09/829,079

AUSTRALIA

Patent Office
Canberra

I, MARIA LEWIS, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PQ 7247 for a patent by CANON KABUSHIKI KAISHA filed on 02 May 2000.

I further certify that pursuant to the provisions of Section 38(1) of the Patents Act 1990 a complete specification was filed on 05 April 2001 and it is an associated application to Provisional Application No. PQ 7247 and has been allocated No. 35026/01.

WITNESS my hand this
Nineteenth day of April 2001

M. Lewis.

MARIA LEWIS
TEAM LEADER EXAMINATION
SUPPORT AND SALES

S&F Ref: 447627

ORIGINAL

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION FOR THE INVENTION ENTITLED:

Printing Using Secure Pickup

Name and Address of Applicant:

Canon Kabushiki Kaisha, incorporated in Japan, of 30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, 146, Japan

Names of Inventors:

Robert Smart and William Simpson-Young and Sue-Ken Yap

This invention is best described in the following statement:

PRINTING USING SECURE PICKUP

Technical Field of the Invention

The present invention relates generally to the field of remote printing, and in particular, to security thereof.

Background Art

The current proliferation of local area networks (LANs) provides users thereof with many conveniences, not the least of which is an ability for a user population, say on a floor of a building, to access a limited number of relatively expensive peripheral devices, eg colour printers.

A problem with use of shared peripherals arises, however, when data to be printed is confidential to the person requesting the print job. If, for example, the printer is remote from the aforementioned person, eg on the other side of a building floor, then a confidential document will typically emerge from the printer before the print job originator arrives at the printer, possibly compromising the confidentiality of the job.

Disclosure of the Invention

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to a first aspect of the invention, there is provided a method of conducting a secure process between a first device and a second device, said method comprising steps of:

inputting security key information into the first device;

establishing said process between the first device and the second device;

suspending the process at a stage prior to completion thereof; and

enabling the suspended process to proceed to completion dependent upon inputting corresponding key information into the second device.

According to another aspect of the invention, there is provided an apparatus for implementing the aforementioned method.

According to another aspect of the invention there is provided a computer program product including a computer readable medium having recorded thereon a computer program for implementing the method described above.

The invention has the significant advantage of supporting secure remote printing by a mix of physical and electronic, public and private keys.

Brief Description of the Drawings

A number of preferred embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1 illustrates a system in which system embodiments of the invention can be practiced;

Fig. 2 depicts devices connected in a system according to a preferred embodiment of the invention;

Fig. 3 illustrates minimum and backward compatibility between devices in the system of Fig. 2;

Fig. 4 shows exemplary network protocols used in the system of Fig. 2;

Fig. 5 illustrates direct acquisition of a target device according to a preferred embodiment;

Fig. 6 shows "pull service provision" using self-generating chaining for a single interposing device according to a preferred embodiment;

Fig. 7 shows "push service provision" using self-generating chaining for a single interposing device according to a preferred embodiment;

Fig. 8 shows "pull service provision" using local self-generating chaining for a plurality of interposing devices according to a preferred embodiment;

Fig. 9 shows "pull service provision" using centralised self-generating chaining for a plurality of interposing devices according to a preferred embodiment;

Fig. 10 illustrates a flow process for self-generating chaining according to a preferred embodiment;

Fig. 11 depicts establishment of a communication session with a legacy device in the preferred embodiment;

Fig. 12 illustrates incorporation of security features in the preferred embodiment;

Fig. 13 illustrates use of a separate local display by an initiating device having a limited display capability in the preferred embodiment;

Fig. 14 illustrates equivalence of devices and services in the preferred embodiment;

Fig. 15 depicts extension of the capabilities of the preferred embodiment to an external network;

Fig. 16 illustrates use of a server with multiple legacy devices in the preferred embodiment;

Fig. 17 shows an example of the preferred embodiment; and

Fig. 18 shows detail for a GUI in the example of Fig. 16.

Detailed Description including Best Mode

In the context of this specification, the word "comprising" means "including principally but not necessarily solely" or "having" or "including" and not "consisting only of". Variations of the word comprising, such as "comprise" and "comprises" have corresponding meanings.

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

Fig. 1 illustrates a system including a network 1706 in which embodiments of the invention can be practiced. It is assumed for the sake of illustration that the network 1706 supports the Internet suite of communication protocols (ie TCP/IP).

A user has plugged a digital camera 1700, containing a number of digital images stored in internal memory (not shown) into the network 1706, wishing to print the images. Two printers 1702 and 1704 are also connected to the network 1706. In order to print the images there must firstly be awareness by the user, that the printers 1702 and 1704 are connected to and thus accessible on the network. Furthermore, interworking must be possible between the camera 1700 and at least one of the printers 1702, 1704. For example if the images are to be printed with a resolution of 500 dpi (i.e. dots per inch), the printer(s) must support the required resolution.

Fig. 2 depicts a system according to a preferred embodiment of the present invention. A user has connected a camcorder 102 to a network 100, to which are also connected a number of other devices or services. The terms "devices" and "services" are used interchangeably in the specification. The user wishes to print the images in the camcorder 102. Each device 102 – 108 is depicted as having an information type, a profile, and a discovery/announcement process (eg 110, 112 and 126 respectively in relation to the device 102). As noted below, not all devices require discovery and announcement capabilities, and it is accordingly sufficient for certain devices to support an announcement capability only. The information identified as 110 represents the information type and/or the information itself, the intended interpretation being clear from the context.

In the present description, information type is typically declared and/or defined in the device profile 112. In general, devices of interest transmit, receive, or process different types of information. For example, the camcorder 102 records visual images, stores the images in an internal memory, and when so commanded, transmits the stored information on an output line to an external display device such as a TV monitor. A profile 112 characterises the information 110.. Devices such as the camcorder 102 can typically operate in different communication modes. Accordingly, the camcorder 102 can in a first mode of operation, record images in a compressed format, storing the resultant compressed data in internal memory to obtain the longest "filming time". Alternatively, the camcorder 102 can in a second mode, operate in uncompressed mode, storing uncompressed data with higher quality, but at a price of providing less filming time. Each such mode has an associated profile characterising service attributes of the associated information. Therefore, the camcorder 102 has a number of different profiles 112.

Turning to the issue of "network awareness", each of the devices 102-108 has a discovery/announcement capability designated 126-132 respectively. It is not necessary for every device to have both an announcement and a discovery capability, however preferably, devices should have at least an announcement capability.

Service Discovery is the process that lets services discover other nearby services on the network. Remote services, ie. those services described in more detail with reference to Fig. 14, need not be discovered this way. Almost any Service Discovery system, for example the Jini^{TM} discovery service, can typically be used in the present embodiment. The service discovery service used in the preferred embodiment assumes that multicast is configured for the local network and a local-use multicast address is set up to cover the local site.

Network services multicast an announcement as soon as they connect to the network. They also multicast at regular intervals, to handle lost packet situations. Services can also multicast whenever they see a new service appear that they didn't know about. Accordingly, they learn of the new service, and the new service learns about them shortly after connection to the network. Multicast service announcements also list the services seen. Therefore, if a service sees an announcement which doesn't list the recipient, then the recipient knows to reannounce itself.

The Service Discovery process can result in each service learning about other services as soon as both are connected to the network. This immediate discovery is not mandatory, particularly in larger networks. The standard mode of operating for the present embodiment is for every service to get the Service Description of every other service. This is efficient for small local networks, however, in relation to large sites, it is preferable to make use of a Service Description Directory.

Service Discovery is not an appropriate place to transfer significant amounts of information about the service itself (ie the service aspects dealt with in the profile). The basic intent is to tell other services the Universal Resource Indicator (URI) identifying the service. Other services can then contact that service to obtain the Service Description in the profile matching process. Some small amount of profile information may however be exchanged during the discovery/announcement process, thereby reducing the need for every service to get the service description of every other service. In the protocol used in the present embodiment, two pieces of extra information are, therefore, provided, namely the "personality" and "type". "Type" is just a text string such as "printer" to describe the service and indicate a default icon to use if required, as in the browser.

If a service is shut down cleanly before removal from a network then, it will multicast a Service Deletion message. If a service crashes or is otherwise removed from a network without being able to issue a Service Deletion message, then other services will notice that it ceases to send its regular multicast announcement and will delete their reference to the service after testing it.

After devices discover each other, they communicate using the Hypertext Transfer Protocol (HTTP). They can parse an XML message which is sent to their HTTP port using HTTP's "POST" command. A more detailed description of protocols used in the present embodiments is provided with reference to Fig. 8. Examples of XML syntax, commands, data, and service descriptions are found in the Appendix.

As noted above, each device/service has a personality and a type. The "personality" describes its behaviour in the network environment as follows:

- NO_SERVICE. This is used when indicating that a service is being deleted from the local environment.
- PASSIVE. This means that the service will not actively seek to use other services on the local network. As a practical matter a PASSIVE service will not list other services in its service discovery announcements, nor is there any need for other services to ensure that a passive service knows about them.
- ACTIVE. This is a normal network device or service that is interested in other services on the network.

- COMPUTE. This is a service which provides a platform in which new services can be started.
- REMOTE. A remote service provides services that are discovered in other ways than by service announcements. A REMOTE service need not advertise itself locally at all.

The "type" is a text string that describes the service offered in a rough way. Other services can go to the Service Description to get the detailed exact information.

Typical type names include: "printer"; "camera", "browser", "DVC", "DTV", "control", "compute", "service". Any unknown name will be treated as a general undefined "service". The following names in this list deserve particular comment:

- "control" means the device can act as a user control point for some external service. Normally this means at least a Graphic User Interface (GUI) with some sort of pointing device. Voice or text based control are other possibilities. As with printing this doesn't specify what control protocols the device supports: other services need to get the Service Description to find that detail.
- "compute" means a device that can be used to start other services. For example consider a printer needing a conversion service and not having the memory requirements to run it itself. The printer can tell the "compute" service to start the required conversion service.

Returning to Fig. 2, when the camcorder 102 is initially connected to the network 100, the associated discovery/announcement process 126 searches for and identifies other devices which are connected to the network 100, this being performed in cooperation between the respective discovery/announcement processes 126-132 in each device 102-108 as described above. The camcorder 102 thereby discovers the device 104, which is a low resolution printer, by means of respective discovery/announcement processes 126 and 130. The process of discovery/announcement is depicted by the dotted line 134. The camcorder could also discover the low level printer 104 by means of another service, for example browser 146 which is connected to the network 100 by a connection 148.

In the preferred embodiment, XML messaging is the communication protocol of choice, as noted in the description relating to Fig. 4.

An example provided below makes use of discover/announcement protocol options, which include a protocol referred to as "miniSLP".

Like other Service Directory protocols, miniSLP assumes that multicast is configured for the local network and disclosed embodiments use a local-use multicast address set up to cover the local site. In miniSLP the services multicast an announcement as soon as they connect to the network. They also multicast at a regular intervals, but this is only to handle lost packet situations. Services also multicast whenever they see a new service appear that they didn't know about. This means that they learn of the new service and the new service learns about them almost immediately after connection to the network. Multicast service announcements also list the services seen. This is so that if a service sees an announcement which doesn't list the recipient, then the recipient knows to reannounce itself. The result of the Service Directory process is that each service learns about each other service as soon as both are connected to the network.

Service Discovery is not an appropriate place to transfer significant amounts of information about the service. The basic intent is to tell other services the URL identifying the service. Other services can then contact that service to obtain the Service Description. However some small amount of information reduces the need for every service to get the service description of every other service. In the case of miniSLP two pieces of extra information are provided: the personality (one of a small number of possibilities describing behaviour) and type, which is just a test string such as "printer" to describe the service and indicate a default icon to use if required, in the browser.

__Example 1: XML sequence illustrating discovery/announcement__

When the browser is informed (via the service discovery protocol) that the camera has come on-line, it asks for the camera to send its description, in a sequence as follows:

(i) A camera comes on-line.

(ii) MiniSLP or Jini or other suitable discovery &/or announcement protocol informs a browser device of the presence & address of the camera.

(iii) The browser sends an update request (`describe` command) to the camera using the following XML code fragment:

```
<Message from="http://host/browser"
     to=http://host/camera                                    [1]
     time="Tue Mar 23 14:12:10 EST 1999">
<describe/>
</Message>
```

(iv) The camera then answers with its description as follows:

```
< Message from="http://host/camera"
     to="http://host/browser"
     time="Tue Mar 23 14:12:11 EST 1999">
<update>
<device url="http://host/camera"
     xmlns="http://dochost//"
     xmlns:camera="http://dochost//camera.html">
<identity name="PowerShoot" class="camera"
     manufacturer="Canon" model="PS350"
     owner="CMIS" version="1.3c"/>
<location slot="38a" room="141" floor="1" building="e6b"/>
<comment comment="Demo camera"/>
<content>
     <camera:thumbnails base="thumbs/" refbase="images/"
          start="301" end="308"
          suffix=".jpg" format="JPEG"/>                      [2]
     <camera:images base="images/"
          start="301" end="308"
          suffix=".jpg" format="JPEG"/>
</content>
</update>
<commands>
     <getData><content><camera:thumbnails/></content></getData>
     <getData><content><camera:images/></content></getData>
</commands>
</Message>
```

The browser can then display this description directly, or wait for the user to click on the camera icon or the browser might take some other action with this newly-obtained camera description.

The discovery/announcement processes of all devices 102-108 connected to the network 100 are sufficiently standardised and compatible to support a minimum level of communication to be established. Aspects of minimum compatibility and backward compatibility are described in further detail in regard to Fig. 3. A device can, in general, search for and discover other devices in a number of different ways. Fig. 2 illustrates a method based on cooperative search and discovery by all devices. Alternatively, a device need not actively "discover" other devices, but can instead be a passive "listener", being informed by other devices of their presence. Furthermore, a registry server can provide a Service Description Directory service which is accessible to other devices and services on the network.

Fig. 3 addresses the aspects of minimum compatibility, and also forward and backward compatibility. This applies to both the discovery/announcement process, and to the profile matching process. In the description below in relation to Fig. 3 only therefore, the term "information" relates both to the information feature 110 and to the discovery/announcement process 126 in device 102 in Fig. 2. Considering a number of different devices connected to a network, a table 1200 describes the relationship between the different devices in terms of their information type (in a column 1202), and their information version (in a row 1204). Different information types, in different versions, will generally be present in a system. For example, a device X (designated 1212) supports information type 1 (designated 1206) in version 1 (designated 1208). A device Y (designated 1214) also supports information type 1 (ie 1206) however in version M (designated 1210). Version 1 represents the earliest and/or the most basic version of a particular information type. The present embodiment provides that all devices supporting, for example, information type 1 (ie 1206), irrespective of the information version which they support, can establish communications at least at the version 1 level. Therefore, all devices supporting information type 1 (ie. 1206) can communicate at least at the version 1 (ie. 1208) level. Furthermore, backward compatibility is provided to some extent, which means that a first device supporting information type 1 in version M and a second device supporting information type 1 in a later version N are, despite the apparent version incompatibility, nonetheless often able to communicate at the version M level.

Fig. 4 illustrates a preferred embodiment of the present invention, providing more detail about the individual elements in the system, and in particular, the preferred network protocols. The preferred protocol suite is, however, merely illustrative, and other protocol suites can be utilised. The present embodiment uses the Internet Protocol and related standards as detailed in the Internet Engineering Task Force specification RFC 1122 "Requirements for Internet hosts-communication layers". The required network protocol is defined in the specification "Hypertext Transfer Protocol-HTTP/1.1" RFC 2068. One or more devices 800 and/or one or more services 802 are connected by physical connections (not shown, but which may be a mixture of wired and/or wireless connections) to a network 804. The network protocol is TCP/IP. Typically, the service 802 has an associated profile 806 which provides a description of the characteristics and attributes of the service 802.

Network layer connection to the network 804 by a service 802 makes use of the DHCP protocol 808 (refer to RFC2131 "Dynamic Host Configuration Protocol"). The discovery/announcement process by which services and devices find out about each other is performed using the miniSLP protocol 810 (refer "Service Location Protocol RFC2165" and previous description of miniSLP). Information exchange between a device 800 and the service 802 depicted by the line 812 is performed using the HTTP protocol, while structured information exchange is performed using the XML protocol. For further detail about HTTP, refer to "Hypertext Transfer Protocol- HTTP/1.1", RFC2608. For further information on XML, refer to "Extensible Markup Language (XML World Wide Web consortium recommendation REC-XML-19980210". HTTP/XML communication is made up of commands 814, messages 816, and service/device descriptions 818.

Turning back to Fig. 2, once all the devices 102-108 have discovered each other, thus having established network awareness between the camcorder 102 and the other devices 104 – 108, the next issue is interworking compatibility. The profile processes 102 and 138, which as noted are associated with respective information 110 and 136, establish whether the desired print job can be performed. As described in relation to Fig.

3, the profile processes 112, 138, 140 and 144 for all the devices 102-108 connected to the network 100 are sufficiently standardised to be capable of at least a minimum degree of communication. Therefore, an initiating device, eg the camcorder 102, can identify the population of devices connected to the network, and is, at least to some extent, aware of their attributes and capabilities. The user of the camcorder 102 is therefore aware, by means of the discovery/announcement process previously discussed, of the low resolution printer 104 which is attached to the network 100. The user is also aware, by virtue of the "personality" and "type" information received during announcement/discovery, of basic service attributes of the printer 104. This information may be sufficient to initiate a print command immediately, however in general, exchange of service description information is required. Therefore, when the user of the camcorder 102 commands the camcorder 102 to print the stored images 110 on the printer 104, the following process takes place. In the first instance, communication depicted by a line 114 is established between the profile processes 112 and 138 which are associated with the camcorder 102 and the printer 104 respectively.

Thus, for example, if the information 110 in the camcorder 102 is desired to be printed at 500 dpi, (dot per inch) and the printer 104 is able to print at the desired resolution, then the respective profile processes 112 and 138 conclude that the two devices 102 and 104 are compatible. Once the aforementioned "match" between the respective profile processes 112 and 138 is established, communications as designated by the line 116 is set up between the camcorder 102 the printer 104 respectively, and the images in the camcorder 102 are printed on the printer 104.

An XML messaging sequence, incorporating the browser 146 into the process, is now described. Firstly recall that the camcorder 102 has announced itself and sent profile information to the browser according to the XML code fragments [1] and [2] provided above. The printing process continues with the user selecting a "thumbnails" icon in the camera description, after which the browser sends the selected element to camera. The browser knows that `thumbnails` can be selected because of the `commands` element in the camera description.

Example 2: Sequence of a browser retrieving thumbnail images from a camera

(i) The browser sends a getData request for one or more elements to the camera.

```
<Message from="http://host/browser"
     to="http://host/camera"
     time="Tue Mar 23 14:13:05 EST 1999">                         [3]
<getData><camera:thumbnails base="thumbs/" refbase="images/"
start="301" end="308" suffix=".jpg" format="JPEG"/></getData>
</Message>
```

(ii) The camera answers as follows, noting that along with each thumbnail image, the camera encapsulates a command which is to be used to retrieve the associated reference image (<getData>). This command will be typically retained by the browser and associated with each retrieved thumbnail image.

```
<Message from="http://host/camera"
     to="http://host/browser"
     time="Tue Mar 23 14:13:06 EST 1999">                         [4]
<data>
     <image format="JPEG" name="thumbs/301.jpg">
          <BASE64>image data</BASE64>
          <getData format="JPEG" name="images/301.jpg"/>
     </image>
     <image format="JPEG" name="thumbs/302.jpg">
     etc. (8 images)
</data>
</Message>
```

The browser need not have any understanding of the command because the command is intended for the camera, not the browser. The browser need merely send the command back to the camera at an appropriate point, e.g. when a user selects a thumbnail for viewing, the browser could interpret that action as requiring it to send the associated getData command back to the camera, which then yields a larger version of the image for viewing. These associated commands sent with data to a host device or service can also include graphical user interface (GUI) elements and attributes. A GUI could be implicitly built around retrieved data and associated commands inside a host that need not have any comprehension of the commands or associations (the GUI elements and attributes will typically be understood by the host). It is even possible to link multiple devices or services and their various states or conditions into controlling the execution or transmission of associated commands, as well as introducing procedural or scripted code. These facilities can be supported without requiring the host device or service to need any understanding of the potential actions it might be hosting.

When the browser gets the thumbnail image data from the camera, it typically displays it in its content pane (eg see Fig. 18). There is no "print" command per se in the present embodiment, however instead the printer receives a `<data>` command that it interprets as a print command. Instead of sending the actual image, the browser sends an image reference to the printer. The browser can send to the printer the command to get the image from the camera (`<getData>`).

Example 3: Sequence of a browser directing a printer to print an image from a camera

The user typically selects a thumbnail image displayed in the browser window and requests to print the related full size image by any reasonable means (including implicit means within the selection process). The browser then sends the camera's `getData` (or other) command associated with that thumbnail image to the desired printer. It is assumed that the printer and its capabilities or attributes were previously identified to the browser. The camera and its capabilities or attributes might or might not have been previously identified to the printer.

(i) The browser sends an image reference to the printer.

Instead of sending the actual image, the browser sends to the printer the associated command to get the image from the camera (`<getData>`). The browser need not have any understanding of the associated command (`getData`) it has sent to the printer.

```
<Message from="http://host/browser"
     to="http://host/printer"
     time="Tue Mar 23 14:14:01 EST 1999">                   [5]
<data url="http://host/camera">
     <getData format="JPEG" name="images/301.jpg"/>
</data>
</Message>
```

(ii) The printer then sends a request for the image to the camera (identified by the name space or URL in the command received by the printer).

The printer need not have any understanding of the command (`getData`) it sends to the camera.

```
<Message from="http://host/printer"
    to="http://host/camera"
    time="Tue Mar 23 14:14:02 EST 1999">                    [6]
<getData format="JPEG" name="images/301.jpg"/>
</Message>
```

(iii) The camera sends the image to the printer.

```
<Message from="http://host/camera"
    to="http://host/printer"
    time="Tue Mar 23 14:14:03 EST 1999">
<data>
    <image format="JPEG" name="images/301.jpg">             [7]
    <BASE64>image data</BASE64>
    <getData format="JPEG" name="images/301.jpg"/>
    </image>
</data>
</Message>
```

When the printer receives this last message, it sees that this time it has received actual data, and will print it. Note that more interaction with the user could be requested by the printer, by sending messages to the browser for example, to pop up a dialogue asking for number of copies, etc.

Communications of this type can be embedded into XML documents as references, or as URIs to data which is required to be included, before an operation can be completed. For example, an XML document intended for printing can reference camera images, using the message syntax indicated. The document also contains textual content for rendering. The image references can be resolved by the preceding method, prior to rendering.

Each network service has a service description used in the profile matching process. This is an XML file describing fairly static information about the device. See Tables A4 to A6 in the Appendix for more details and examples. Explicitly the description covers information that changes on human time scales: seconds or preferably longer. There is a different file called status that contains more rapidly changing information.

The information in the Service Description is intended to allow other services to plan how to use the service. Examples of information that might appear:

- Supported resolution.
- Physical location
- Supported commands (with version number)
- Currently loaded paper tray
- Available images
- Supported image formats

Service Description information is available from at least four sources as follows:

- Static information about the hardware or software provided by the manufacturer.
- Information that is entered and updated by the owner of the device: location; access control; owner's public key;....
- Information that the service generates during its operation by examining its own internal state: loaded paper; error status; images or video available; ...
- Information that relates to the capabilities of the machine: personality; type; supported commands.

One service can ask another to notify it when the Service Description changes. This is particularly useful for a service like the browser which presents information about the state of the service to a user. The user can see what paper is currently loaded in the printer or where the scanner is plugged in today.

The Service Description of a printer specifies that it understands, for example, the Line Printer Daemon protocol (LDP), IETF RFC 1179. Another service can use that to send print jobs to the printer. The service communication protocol uses XML over HTTP.

Each service is identified by a Universal Resource Indicator (URI), just like a web page on the WWW^{TM}. Just as on the WWW^{TM}, the preferred embodiment uses HTTP for communication. HTTP has three basic communication mechanisms:

- GET. This is used when you click on a link in a web browser: the browser GETs the page and displays it.
- POST. This is commonly used on the web when you fill in a form and press "submit". The form data is sent and the web server returns a replacement page which typically reports the success of your action.
- PUT. This is sometimes used for file uploads.

The present embodiment uses POST. However instead of sending web form data it sends an XML file that describes a network command. The response it gets back is not a displayable web page but another piece of XML giving a response to the command. Frequently the response just indicates success or failure, but complex data can be returned.

XML is a way of structuring information that looks very like HTML used in web pages. In fact well formed HTML is substantially a subset of XML designed to describe how to layout text and other information on a browser page.

Profile processes, including aspects of compatibility establishment, can be implemented in a number of ways, depending on system priorities. A simplest process is to seek an exact match with a profile name or type. Another option is to pre-grade partial matches (eg an exact resolution match but not a print-size match). A further option is to create profiles of profile matches, ie meta-meta data describing the extent to which partial matches are preferable, one over another.

A number of criteria in accordance with which compatibility can be established are presented in the following table. Many options exist for how a device or service or a consulted discovery or directory service might select a suitable intermediate service or services. The options can be selected statically or dynamically based on the priorities of the system or its implementors.

Intermediate Service Selection Criteria	**Explanation**	**Issues/Examples (and/or combinations are possible)**
Static priority	Selection decisions are based on permanent criteria.	I/O matching, quality-matching, etc.
Dynamic priority	Selection decisions can change. e.g. could be based on the job details, user requirements, network/service activity level, etc.	Job or media cost, speed, processing requirements, storage requirements, quality, etc.
I/O Matching	Device/service attributes are compared. The simplest model is to compare only the nearest port in the stream, i.e. don't check the service's I/O function, only its nearest interface.	This simple method can have drawbacks such as not controlling the length of a chain (or the time to build it); no guarantee of finding the best chain or any complete chain; potential loss of quality or increase in job processing time with added intermediate services.
Function & I/O matching	Device/services are matched based on their nearest port in the stream and another property such as function or other port matching, etc.	This method provides a better guarantee of matching a service to the local needs. It still does not guarantee completion of a chain (e.g. if a required service is missing then the constraints might be too tight to find an alternative & more indirect chain)
Semi-random matching or annealed matching, etc	Device/services are matched based on partially constrained data with the level of constraint varying under some conditions.	This method might allow more optimal selection of a best-match chain or a near, best-match chain. Many methods are possible, and the intent is to overcome the possibly undesirable effect of local minima in the 'chain space' of available services.
Directory or Discovery services	Provision of lists of available services, optionally along with additional information suggesting best fits for	A service can collect and/or provide information about the network to any other service. It can also be used in a global management

	current job criteria (speed, cost, quality, etc).	capacity to allow some global control of the inherently local nature of chain-building.
Grouped services	Provision of lists or service structures or conglomerates in which a single service is identified but in reality the service is a conglomerate.	A recommended strategy for any stable network for providing optimal chain-building for predictable jobs with typical cost, quality and speed criteria. The localised chain-building method need not know or care how a service conglomerate is constructed or what it is constructed of because only its interfaces and external functional description & attributes are relevant in the chain-building process.
Pre-selection or manual specification.	A device or service could have a preset chain description encoded.	This hard-coded chain-building method can fail easily if a specified device is not available. A less brittle approach is possible if some alternatives or only partial preselection is hard-coded, giving the chain-building operation some limited choices.

Table 1

Devices always establish enough communication to obtain information on operations supported, and associated versions. This is typically performed using a "get description" operation in XML, this operation fetching a description, in XML, of the device. The camcorder 102 has discovered the printer 104 which is, in fact, a "high resolution display". The camcorder 102 does not know about such devices, however a "device description" in the high resolution display says that the display is equivalent to a printer 104. This tells the camcorder 102 enough to be able to use the printer. The camcorder 102 sees that the printer 104 supports the "system.data" command version 7. The camcorder 102 knows how to use the "system.data" command version 6. Since, in the present case, backward compatibility is provided, the camcorder 102 knows that it can send the data to be "printed" using the "system.data" command version 6. Guaranteeing backward compatibility with all previous versions is not a preferred solution, and neither is forcing all future versions to be compatible with all previous versions. A compromise solution is to provide an "escape" compatibility option, whereby for example compatibility is guaranteed in respect of a limited number of major previous versions 1.1, 1.6 and 2.2, but not with 3.2. Alternatively, interposing an intermediate device to provide a "compatibility chain" can be performed, as described below. Pre-defined system commands are designated by a "system." prefix, this indicating that the aforementioned "system.data" commands belong to the pre-defined system command class. In contrast, user-defined commands are designated by a "user." prefix.

The aforementioned description in relation to Fig. 2 related to establishment of direct communication between an initiating device, in this example the camcorder 102 and a target device which in this case is the printer 104. In this example, therefore, the camcorder 102 has succeeded in itself identifying a desired target device, in this case the printer 104.

Fig. 2 further depicts a more complex scenario in which the camcorder 102 has also discovered that a high resolution printer 106 is available on the network 100. In addition to the compressed and normal recording modes previously described, the camcorder 102 is further capable of a high resolution recording mode using compressed data. The user wishes to make use of this high resolution printer 106, however when respective profile processes 112 and 140 of the camcorder 102 and high resolution printer 106 communicate, as depicted by the line 118, it is discovered that the information 110 in the camcorder 102 cannot, according to the information type 142, be directly processed by the high resolution printer 106. This is because the printer 106 cannot directly input the high resolution compressed data 110.

At this point, however, the high resolution printer 106 does not merely reject the print job sent from the camcorder 102. Instead, the printer 106 directs a self generating chaining process by which it searches for other devices on the network which it can interpose between itself and the camcorder 102, providing the necessary interworking if this is possible. The printer 106 identifies a decompressor device 108 which is also present on the network 100. This is an example of "pull service delivery", as against "push service delivery", both of which concepts will be described in further detail in relation to Figs. 5 to 7. Taking into account the previous communication between the profile processes 112, 140 associated with the camcorder 102 and high resolution printer 106, subsequent communication between the respective profile processes 140 and 144 of the high resolution printer 106 and the decompressor device 108, are sufficient to establish that interworking is possible by use of a tandem communications arrangement. In the present "pull" example, the printer requests the decompressor to fetch data from the camcorder. This request is nested inside the request from the printer to the decompressor for the decompressed data, as will be described in the XML code fragments provided in Example 4. It is noted that Example 4 makes reference to a codec instead of a decompressor. The tandem arrangement entails the camcorder 102 sending the print job depicted by information 110 to the decompressor 108. The decompressor 108 decompresses the received information 110 and sends the decompressed information to the high resolution printer 106 which is able to print this uncompressed information. The tandem communications set up between the camcorder 102 and decompressor 108, and between the decompressor 108 and the high resolution printer 106 are depicted by dashed lines 122 and 124 respectively.

Figs. 5 to 7 provide a more general description of the self generating chaining process used in the present embodiment. Fig. 5 illustrates the establishment of direct communication between an initiating camcorder 1000 and a target low resolution printer 1006. The camcorder 1000 wishes to print information 1002, which is low resolution image information. The information 1002 has a profile "X" designated as 1004. The camcorder 1000 discovers a low resolution printer 1006 by means of a discovery/announcement process depicted by a dashed arrow 1010. The camcorder 1000 and printer 1006 exchange/negotiate respective profile information 1004, 1008 as depicted by a solid arrow 1012, concluding that both devices support profile feature X. It is noted that the profile 1008 of the printer 1006 also is "X" as shown in the figure. Thereafter, the camcorder 1000 and low resolution printer 1006 establish communications as depicted by an arrow 1014 whereby the camcorder 1000 transmits its low resolution picture information 1002 to the low resolution printer 1006, which in turn prints the desired output. This is an example of direct acquisition by the camcorder 1000 of the desired target low resolution printer 1006, in this case, both devices having a common profile feature "X" designated as 1004, 1008 respectively. The profile features 1004, 1006 represent information types/versions as previously described in relation to Fig. 3, where the aspects of minimum and backward version compatibility were described.

Fig. 6 shows the camcorder 1000, which in this case contains high resolution picture information 1112, which is associated with the profile feature "Y" (ie. 1114). The camcorder 1000 or an additional device (not shown), discovers, as depicted by the dashed arrow 1010, the low resolution printer 1006, which has a profile feature "X" (ie. 1008). The camcorder or other device (eg. a browser), commands the printer to retrieve picture information from the camcorder (in a "pull" service provision approach). At this juncture, there is an apparent mismatch between the camcorder 1000 and the low resolution printer 1006. However, the printer 1006 initiates a "self generating chaining 'pull' process" by which it discovers, as depicted by the dashed arrow 1104, a "resolution conversion device" 1108, which is capable of being interposed between the profile feature Y (ie. 1004) and the profile feature X (ie. 1008).

Criteria provided in Table 1 above can be used to establish chaining priorities and approaches. The device 1108 has an interworking profile features "Y→X" (ie. 1108), which can be considered as having a "Y" profile at one interface, and an "X" profile at another interface. Therefore, the device 1108 can interwork between the two other devices 1000, 1006 having respective profiles "Y" and "X". The "Y" profile of the camcorder 1000 thus matches the "Y" profile of the resolution conversion device 1108, and the "X" profile of the resolution conversion device 1108 matches the "X" profile of the printer 1006. The low resolution printer 1006 ascertains the profile feature 1110 by a process depicted by the arrow 1102. Having ascertained, by means of the self generating chaining process described, that the resolution conversion device 1108 can be interposed between the camcorder 1000 and the printer 1006, the printer directs the camcorder 1000 and the resolution conversion device 1108 to establish communications denoted by the arrows 1106 and 1100, thus enabling the high resolution picture information 1112 to be transmitted to the resolution conversion device 1108 as depicted by a line 1106, after which the converted information (converted from high resolution to low resolution suitable for the printer 1006) is transmitted to the low resolution printer 1006 as depicted by a line 1100. The description provided above is directed to high and low resolution printers. The following example addresses a similar situation, however the example is directed in particular towards images in two different formats, namely a first format conforming to the Joint Photographic Experts Group (JPEG) standard, and secondly, to a bit-mapped format (BMP).

Example 4: XML code sequence of pull service provision with camera/printer mismatch.

In an extension to Example 3 in which a browser directs a printer to retrieve and print an image from a camera, the printer discovers that it does not have an input format compatible with the output format of the camera. It is assumed that the printer and its capabilities or attributes were previously identified to the browser. If the camera's capabilities or attributes were previously identified to the printer, then the printer will be able to compare its input formats with the camera's output formats and decide if a direct match is available (in this example a direct match is not available). Alternatively, if the camera's capabilities or attributes were not previously identified or not previously fully identified to the printer then the printer might interrupt the example command sequence below with a describe request to either the camera or a similar information request about the camera to an announcement/discovery service (if available). Thereafter, the printer will make the comparison of input and output formats and decide if a direct match is available. The associated code sequence is as follows:

(i) The browser sends an image reference to the printer.

Instead of sending the actual image, the browser sends to the printer the associated command to get the image from the camera (`<getData>`). The browser need not have any understanding of the associated command (`getData`) it has sent to the printer.

```
<Message from="http://host/browser"
     to="http://host/printer"
     time="Tue Mar 23 14:14:01 EST 1999">
<data url="http://host/camera">                              [8]
     <getData format="JPEG" name="images/301.jpg"/>
</data>
</Message>
```

(ii) The Printer decides if a direct I/O format match is available with the camera.

The printer need not have any understanding of the command (`getData`) it sends to the camera but it would be typical for a device or service to check the format attribute value in the `getData` command for a match with its own input format capabilities. In this example, the printer's only input format capability is bitmap (BMP) and so the printer recognises a failed direct match between its input format and the camera's output format. The printer accordingly does not send the following XML code fragment:

```
<Message from="http://host/printer"
     to="http://host/camera"
     time="Tue Mar 23 14:14:02 EST 1999">                      [9]
<getData format="JPEG" name="images/301.jpg"/>
</Message>
```

(iii) The printer instead requests/identifies an intermediate service.

The printer either consults its own record of available services, or consults another service (typically a discovery &/or announcement service or directory service) for a list of available services that might operate as an intermediate step between itself and the camera. There are many options at this point for how the printer or a consulted discovery or directory service might select a suitable intermediate service or services. The options can be selected statically or dynamically based on the priorities of the system or its implementors. Table 1 illustrates examples of intermediate service selection options. The table also describes a role of a directory service in chaining.

The printer accordingly identifies a service that has a `bmp` output format compatible with the printer's input format. In this example the printer makes no check of the selected intermediate service's input format capabilities. This behaviour is reasonable for a printer that might normally have limited capability (for cost-reduction purposes). More capable selection and reasoning capability concerning intermediate services might be provided in other services or devices. In this example the identified intermediate service is labelled as "codec" with a URL of http://*host*/codec.

(iv) The printer sends the camera image request to the intermediate service.

The printer wraps the camera image request inside a request to the identified intermediate device.

```
<Message from="http://host/printer"
     to="http://host/codec"
     time="Tue Mar 23 14:14:03 EST 1999">
<getData format="BMP">                                          [10]
     <data url="http://host/camera">
          <getData format="JPEG" name="images/301.jpg"/>
     </data>
</getData>
</Message>
```

The printer has requested a BMP format image from the codec. The printer has not specified a URL or filename within the codec for the origin of the file it wants, but rather has provided a `data` statement containing a command for the camera.

(v) The codec sends the image request to the camera.

The codec service need not understand the contents of the `data` statement, but only need forward it to the specified destination (the camera) and await the return of data.

```
<Message from="http://host/codec"
     to="http://host/camera"
     time="Tue Mar 23 14:14:04 EST 1999">                        [11]
<getData format="JPEG" name="images/301.jpg"/>
</Message>
```

(vi) The camera sends the image to the codec.

```
<Message from="http://host/camera"
     to="http://host/codec"
     time="Tue Mar 23 14:14:05 EST 1999">
<data>
     <image format="JPEG" name="images/301.jpg">                [12]
     <BASE64>image data</BASE64>
     <getData format="JPEG" name="images/301.jpg"/>
     </image>
</data>
</Message>
```

(vii) The camera returns the requested data to the codec and also (optionally) associates a getData command with the image.

(viii) The codec processes the job and sends this to the printer.

The codec has an implied processing job to complete because of the discrepancy between the input and output formats of the data provided to it and requested from it. The codec fulfils its obligation by understanding the implicit conversion command from the printer, and accordingly converts the camera's JPG format image to a BMP format image requested by the printer. Such implied commands can be controlled in more detail by external devices by attributes and parameters in the connecting device's/services interfaces or in explicit attributes or parameters in the commands and data provided to an intermediate device or service. Furthermore, explicit processing commands can be provided to an intermediate device or service.

The codec returns the requested & processed camera data to the printer and also (optionally) associates a getData command with the image (see [13a] or [13b]).

```
<Message from="http://host/codec"
     to="http://host/printer"
     time="Tue Mar 23 14:14:06 EST 1999">
<data>                                                          [13]
     <image format="BMP" name="images/301.bmp">
     <BASE64>image data</BASE64>


     <getData format="BMP" name="images/301.bmp"/>             [13a]
     OR
     <data url="http://host/camera">
          <getData format="JPEG" name="images/301.jpg"/>       [13b]
     </data>
```

```
        </image>
    </data>
    </Message>
```

The example above optionally shows the codec adopting the camera's image pathname but changing the image type. This new codec pathname is then provided as a handle to the printer. The codec is under no obligation to retain any similarity between its image path name and the camera's image path name. However there are benefits if a predictable naming similarity is retained, especially if unrelated commands can be multiplexed then the printer might need naming similarities to aid it in sorting which response belongs to which original request. One solution would be for the codec to return its arbitrary image pathname to the printer and provide an attribute indicating the original pathname (and possibly device) from which the printer requested the image.

The associated getData command within the image tags could be returned from the codec to the printer in a number of ways. For instance, if the codec retains a cache of the converted image then it might replace the camera's associated command with its own version (see XML code [13a]), thereby removing the camera's involvement from any future request for that image. Alternatively, the codec might wrap the camera's associated `getData` request inside a `data` command identifying to which device it should be directed if transmitted in future (see XML code [13b]). Therefore, in future the printer might send the associated `data` command with nested `getData` command to the codec and the codec would be able to relay the `getData` command to the camera.

The above example can be extended substantially indefinitely where the codec can represent any number of devices or services involved in the pull chaining process between printer and camera. As each device or service is enrolled into the chain, the command(s) originally sent by the printer to the camera can be nested inside a wrapper intended for the next device or service in-line. Similarly, any associated `getData` command included by the camera in its returned data can be wrapped by each device or service on its way back to the printer. This wrapping or nesting will preserve the shape &

details of the chain for the printer if it desires to activate the camera's associated command. The passage of the associated command from the camera back to the printer, if nested by each intermediate device, provides a description of the complete chain to the printer, allowing expedited reactivation of the chain at a subsequent time.

Fig. 7 illustrates a "push service provision" embodiment, in contrast to the "pull service provision" embodiment described in relation to Fig. 6. In Fig. 7, after the camcorder 1000 discovers and characterises the printer 1006, concluding that a profile mismatch exists, the camcorder 1000 discovers the resolution conversion device 1108 by a discovery announcement process depicted by a dashed arrow 1200. The camcorder 1000 ascertains the profile features 1110 of the resolution conversion device 1108 by a communication 1202, after which the camcorder 1000 directs the resolution conversion device 1108 and the printer 1006 to establish communications depicted by lines 1206 and 1208. Thereafter, the camcorder 1000 sends the high resolution picture information 1112 to the resolution conversion device 1108 as depicted by 1206, whereafter the information, having been resolution converted, is sent as depicted by 1208 to the printer 1006. An XML code example of push service provision is now provided.

Example 5: XML sequence of push service provision for camera/printer mismatch.

The camera has previously decided that a direct connection with the printer is not possible and has discovered the codec has a suitable JPEG input format. Similar discussion and explanation as provided in regard to Example 4 applies to the push-chaining sequence in this example.

(i) The camera sends an image reference to the codec.

```
<Message from="http://host/camera"
     to="http://host/codec"
     time="Tue Mar 23 14:14:00 EST 1999">
<data url="http://host/printer">                              [14]
     <image format="JPEG" name="images/301.jpg">
     <BASE64>image data</BASE64>
     <getData format="JPEG" name="images/301.jpg"/>
     </image>
</data>
</Message>
```

(ii) The codec processes the image, and then sends the image to the printer.

```
<Message from="http://host/codec"
     to="http://host/printer"
     time="Tue Mar 23 14:14:01 EST 1999">
<data>
     <image format="BMP" name="images/301.bmp">
     <BASE64>image data</BASE64>                              [15]


     <getData format="BMP" name="images/301.bmp"/>
     OR
     <data url="http://host/camera">
          <getData format="JPEG" name="images/301.jpg"/>
     </data>


     </image>
</data>
</Message>
```

In this example, the codec output format, ie. BMP, is expected. This is because either BMP is the only function provided by the codec, or alternatively, because the codec will check the codec interface with the printer. It is also possible for the camera to imply, or alternatively, explicitly set an output format or a conversion function within the codec. One method of doing so is to add a file format attribute to the host printer URI which is provided in the fourth line of the code fragment with the reference numeral [14]. Accordingly, the host printer URI could take the alternate form

<data url = "http://host/printer" format = BMP>.

This implies a conversion process, or an interface selection, to the codec.

The "pull service provision" embodiment described in relation to Fig. 6, and the "push service provision" embodiment described in relation to Fig. 7, differ in relation to the entity which initiates and defines the direction of the self generating chaining process. In the pull process, it is the printer 1006 which initiates the required device chain building to service the requirements of the camcorder 1000. In the push process, it is the camcorder itself 1000 which initiates building of the necessary chain of devices.

Two self generating chaining processes are now described. A preferred method is termed "local" chaining, and another chaining method, termed "centralised" chaining is also described.

Fig. 8 depicts a local self generating chaining process applied to a more general case, where a plurality of interposing devices must be found, and a corresponding device chain established. The camcorder 1800 wishes to print compressed high resolution picture information 1802 which has an associated profile feature "Z" (ie. 1804). The camcorder 1800 discovers a low resolution printer 1806 by the discovery announcement process 1818. Profile information is exchanged as depicted by 1820, and a profile mismatch between the camcorder feature "Z" (ie 1804) and the printer feature "X" (ie 1808) is identified.

A pull service provision embodiment, ie directed by the printer 1806, is now described. The printer 1806 searches for, however is unable to discover a single device which, alone, can be interposed in order to establish compatibility between itself (having a profile "X" ie 1808) and the camcorder 1800 with its profile "Z" (ie 1804). Accordingly, the printer 1806 searches for one or more suitable devices with profile "X". Depending on a prevailing selection scheme, the printer 1806 will potentially select one of the discovered profile "X" devices to begin the chain building process. The figure shows that a device C, a resolution converter 1810 is selected by the printer 1806, noting that the resolution converter 1810 has a profile with a conversion feature Y -> X (ie 1812). The printer 1806 passes an "image fetch" command to the resolution converter 1810, which proceeds to build the next link in the chain. This next link is built by the resolution converter 1810 by identifying devices with a compatible "Y" profile, to accord with the "Y" of its own Y -> X profile (ie 1812). The resolution converter 1810 identifies and selects a device D, a decompressor 1814 as being suitable. The decompressor 1814 has a profile conversion feature Z -> Y. The decompressor 1814 now builds the next link in the chain after receiving the "image fetch" command from the printer 1806 via the resolution converter 1810. The decompressor 1814, recognising that its profile is compatible with that of the camcorder 1800, looks no further, and thus completes the chain with the camcorder 1800. This chain completion can be seen by considering the chain as follows:

- the "X" of the "Y->X" profile 1812 of the resolution conversion device 1810 is compatible with the "X" profile 1808 of the printer 1806.
- the "Y" of the "Z->Y" profile 1816 of the decompression device 1814 is compatible with the "Y" of the "Y->X" profile 1812 of the resolution conversion device 1810; and
- the profile "Z" (ie 1804) of the camcorder 1800 is compatible with the "Z" of the "Z->Y" profile 1816 of the decompression device 1814;

The printer 1806 discovered the resolution conversion device 1810 by means of a discovery/announcement process depicted by 1822. Profile feature information was exchanged and negotiated by profile communication depicted by 1824. Similarly, the resolution conversion device 1810 discovered the decompressor device 1814 by means of a discovery/announcement process depicted by 1830. Profile feature information was exchanged and negotiated by profile communication depicted by 1832.

Finally, the printer 1806 directs the camcorder 1800, the decompression device 1814, and the resolution conversion device 1810 to establish communication as depicted by 1826, 1828, and 1836, to support flow of the information 1802 from the camcorder 1800 to the decompression device 1814 as depicted by 1834, from the decompression device 1814 to the resolution conversion device 1810 as depicted by 1828, and finally from the aforementioned resolution conversion device 1810 to the printer 1806 as depicted by 1826.

Fig. 9 depicts a centralised self generating chaining process applied to a more general case, where a plurality of interposing devices must be found, and a corresponding device chain established. The camcorder 1000 wishes to print compressed high resolution picture information 1318 which has an associated profile feature "Z" (ie. 1320). The camcorder 1000 discovers the low resolution printer 1006 by the discovery announcement process 1010. Profile information is exchanged as depicted by 1012, and a profile mismatch between the camcorder feature "Z" (ie 1320) and the printer feature "X" (ie

1008) is identified. A pull service provision embodiment, ie directed by the printer 1006, is now described. The printer 1006 searches for, however is unable to discover a single device which, alone, can be interposed in order to establish compatibility between itself (having a profile "X" ie 1008) and the camcorder 1000 with its profile "Z" (ie 1320). Accordingly, the printer 1006 searches for a pair of suitable devices. The printer 1006 identifies two devices, namely the resolution conversion device 1108 and a decompression device 1300, which in tandem can establish the necessary chain of service interworking. This can be seen by considering the following chain:

- the "X" of the "Y->X" profile 1110 of the resolution conversion device 1108 is compatible with the "X" profile 1008 of the printer 1006.
- the "Y" of the "Z->Y" profile 1302 of the decompression device 1300 is compatible with the "Y" of the "Y->X" profile 1110 of the resolution conversion device 1108;
- the profile "Z" (ie 1320) of the camcorder 1000 is compatible with the "Z" of the "Z->Y" profile 1302 of the decompression device 1300;

The printer 1006 discovered the resolution conversion device 1108 and the decompression device 1300 by means of discover/announcement processes depicted by 1304 and 1308 respectively. Profile feature information was exchange and negotiated by profile communication depicted by 1306 and 1310 respectively.

Finally, the printer 1006 directs the camcorder 1000, the decompression device 1300, and the resolution conversion device 1108 to establish communication as depicted by 1312, 1314, and 1316, to support flow of the information 1318 from the camcorder 1000 to the decompression device 1300 as depicted by 1312, from the decompression device 1300 to the resolution conversion device 1108 as depicted by 1314, and finally from the aforementioned resolution conversion device 1108 to the printer 1006 as depicted by 1316.

Fig. 10 presents a process flow diagram by which the centralised self generating chaining process is performed in a push manner. The process is equally applicable to both pull, and push service provision embodiments (by exchanging actions of source and target devices). Commencing with a process 1400, a source device seeks a suitable target device. If the desired target device is found in a decision step 1402, the chaining process is directed to a further decision block 1404, where the profile compatibility is established. If profile compatibility, either exact or backward is established, the chaining process is directed in accordance with the "YES" arrow from the decision block 1404 to the connection establishment step 1406, and thereafter the chaining process ends at a step 1408. If a suitable target device is not found at the decision step 1402, the chaining process is directed in accordance with the "NO" arrow to the change requirements step 1410 which advises the source device that a change of requirements is in order, whereafter the chaining process returns to the initial process step 1400. If the decision step 1402 identifies a suitable target device, but profile compatibility is not found in the decision block 1404, the chaining process is directed in accordance with the "NO" arrow from the decision block 1404 to a process step 1412, where an index "L", this being the length of the interposing device chain to be sought, is set to 1.

The chaining process is then directed to a process step 1414, which searches for a chain of devices "L" long (L=1 at this stage), which, if interposed between the source and target devices will provide tandem profile compatibility. Thereafter, in a decision block 1416, if an appropriate device chain is found, the chaining process is directed, as indicated by the "YES" arrow from the decision block 1416 to the connection establishment process 1406, and thereafter to the "end" block 1408. If, on the other hand, a suitable device chain with a single interposing device is not found in the decision step 1416, the chaining process is directed in accordance with the "NO" arrow from the decision step 1416 to a process step 1418, where the interposing device chain length, ie. "L", is incremented. Thereafter, a decision block 1420 tests that the length of the interposing chain of devices being sought is not longer than a predetermined maximum length "Q". If the chain is too long in the step 1420, the chaining process is directed, in accordance with a "YES" arrow from the decision block 1420 back to the "requirements change" process step 1410. If on the other hand, the interposing device chain length has not yet reached the maximum length permissible, the chaining process is directed in accordance with the "NO" arrow from the decision block 1420 to the process step 1414, where again a chain of interposing devices is sought with the required tandem compatibility, this time the chain of devices being sought having been incremented in relation to the previous chain which was sought. In this manner, the chaining process flow continues until either a suitable chain of interposing devices is found, or until the permissible length of the interposing device chain is exceeded.

Although the centralised chaining process as described in relation to Figs. 9 and 10 can be used, a decentralised approach can also be adopted. Accordingly, in a pull service embodiment, the printer 1006 can search for only a single interposing device having a profile "X", and demand that that device interface with the camcorder 1000 which has a profile "Z". If the interposing device is unable to interface directly with the camcorder, it repeats the aforementioned process, searching for only a single further interposing device, and demanding that that device interface with the camcorder. This "local" search approach need not be aware of how many links there are in the chain, and from an XML coding perspective, commands are nested inside new commands for the interposing device just found. The flow in Fig. 10 is applicable in so much as each device in the chain successively obeys this flow with L=1 only in order to complete a chain.

Central chaining approaches can provide more optimal chaining solutions, however, they make more demands on devices, and/or on central network control and management. For example, considering Fig. 9, the printer is expected to be able to manage the centralised building of a chain, which would constitute a significant processing load for this device. The local chain building example of Fig. 8 would probably be applied in this instance. Decentralised, or local, control may provide less optimal solutions, and in some cases may even fail to converge on a solution. This approach is, however much easier to deploy. Convergence of a local chaining process is generally guaranteed if (i) the device solution space is constrained (ie there are a finite number of devices in the available pool), and if (ii) a solution exists, (iii) and if corrections can be made to the chain building process.

Turning to Fig. 11, a "legacy" (or "dumb") low resolution printer 200 having an information type 202 is associated with a "smart" "virtual device" 208. This association is depicted by the surrounding box 204. The virtual device 208 has no printing capability, but acts as the "intelligent" front end for the "dumb" printer 200. Thus when the camcorder 102 wishes to print the information 110, the profile processes 112, 210 of the camcorder 102 and virtual device 208 respectively conclude that the print job can proceed, whereafter communication is established as depicted by 206 between the camcorder 102 and the "dumb" printer 200 for printing. In this instance therefore, the intelligence and signalling communication capability are supplied by the virtual device 210, while the actual printing is supplied by the "dumb" printer 200. From a functional perspective therefore, the "dumb" legacy printer 200 has been incorporated into the system by means of the association with the virtual device 208 depicted by the box 204. Typically, the device 208 can provide an information channel (214, 218) and/or a processing step 216 to support the device 200, since the device 200 might be incapable of retrieving data directly from 102.

Fig. 12 depicts use of a physical security key 300 which is inserted into the initiating camcorder device 102, the insertion depicted by a bar 302. The insertion of the security key 300 is accompanied, in some instances, by other control information (not shown). Insertion of the key 300 into the camcorder 102, allows the information 110 to be printed from the camcorder 102 to the printer 104, however, this does not permit the communications session to terminate. Thus, the process does not permit the printer 104 to output the printed pages until the security key 300 is equivalently inserted into the printer 104 as indicated by dashed line 304. This arrangement, termed "secure pick-up", allows secure print jobs to be sent across the network 100, thereby providing the initiating party with the assurance that the print job will not be output by the printer 104 until the print initiator goes to the printer 104 and inserts the security key 300. Printing of the information 110 can be inhibited, requiring the security key 300 to be inserted at the destination printer 104 for enablement, and subsequent printing. Initial insertion of the key 300 is only required once, at setup of a sequence of print jobs, and not every print job. In another embodiment, instead of inserting a physical key 300, a public "inhibition" key can be sent openly to the camcorder 102, and a private "enablement" key embodied physically as the physical object 300, used only at the printer 104. Alternatively, or in addition, the data itself can be encrypted at the camcorder 102 using the public key prior to transmission across the network 100 and decryption at 104 using the private key, 300. This layered security approach provides for both encryption of the data within the network 100, and access security at the printer 104.

Fig. 13 depicts a situation where an initiating device such as the camcorder 102 does not have sufficient display capability to effectively inform the user about the devices and services populating the network 100. In this case, the camcorder 102 discovers a local display 400, the discovery depicted by a dashed line 412. Profile matching between the camcorder profile 112 and the display profile 404 takes place as depicted by a line 410, whereafter if the profiles are found to be matching, either exactly or by backward compatibility, display communication depicted by line 408 is set up between the camcorder 102 and the display 400. In this manner, the user of the camcorder 102 is able, using the relatively limited set of controls available on the camcorder 102, to see a large display 400 which provides convenient and effective representation, and consequently control, of the various network resources so displayed. In the same manner, a control keypad associated with the display 400 can also be used instead of the relatively limited keypad functionality provided by the camcorder 102 itself.

Fig. 14 illustrates how the user of the camcorder 102 utilises a print shop service 508 in order to enhance the quality and attractiveness of the information 110, thereafter printing it on the print shop's high resolution printer 104. The discovery/announcement processes previously described have been omitted for the sake of clarity. The camcorder 102 establishes profile matching with the print shop service 508, as depicted by a line 516. The camcorder 102 also provides the print shop service 508 with the desired service specifications and characteristics during the profile matching process. The print service 508 then performs profile matching with the desired high resolution printer 104 as depicted by line 520. The print shop service 508 having established profile matching, thereby determines that the service desired by the camcorder 102 is feasible, and then establishes communication as depicted by a line 522 with the camcorder 102, pulling the information 110 to the print shop service 508. Thereafter, the print shop 508 processes the camcorder information 110, and also incorporating titling and descriptive textual information from its own archives (not shown).

Once the print shop service 508 completes the processing as specified by the user of the camcorder 102, the service 508 establishes a communication session 526 with the high resolution printer 104, sends the processed information to the printer 104, which produces the desired output. The foregoing description illustrates that services such as the print shop service 508 perform in the same manner as a physical device such as the high resolution printer 104 in the context of the present system. The user of the camcorder 102 can, providing a suitable display is available, view the population of devices (ie. 104) and services (ie. 508), and can formulate a command string or program to perform the set of desired services and actions. The user of the camcorder 102 does not need to adopt a different approach for devices and services. The distributed nature of the system architecture in the present embodiment enables services and virtual devices to reside in any convenient physical location. Indeed, the physical devices and their "intelligent" front ends need not necessarily reside in the same physical location.

Thus, for example, the print shop service 508, can in principle provide an intelligent front-end for all of the printers in the print shop local network, as it may, for instance, do so for the high resolution printer 104, which could be a legacy printer. In the event that the functionality provided by the limited set of camcorder control keys is insufficient, a control device (eg a browser running on a networked PC) can be incorporated. Various options can be used for distributing functionality between the camcorder and the PC control device. For example, the PC can establish a dialogue with the print shop, to be followed by a command from the PC to the printshop to commence a desired process. This command contains job details and a URI for the camcorder. The printshop thereafter initiates a pull process with the camcorder, which supplies the required data.

Fig. 15 illustrates how a remote device 600 which lies beyond a boundary 604 of the system previously described, can be accessed. Providing that the desired remote device, in this case a high resolution printer 600 resides at a known network address (eg. a Universal Resource Identifier or URI), communication as depicted by a line 602 can be set up in order to send desired information 110 from the camcorder 102 to the printer 600. There need not be a discovery/announcement process in this case, neither need there be a profile matching process, and the information 110 is transmitted on the communication path 602 in an open-ended fashion to the printer 600. This mode of operation may or may not assume a knowledge of the printer 600 capabilities, and an error in this regard could result in a failed print job if a profile matching method is not enabled. Security can be added in this scenario using public/private key inhibition/enablement, as described in relation to Fig. 12. Encryption can also be added as described.

Fig. 16 illustrates how the virtual device concept depicted in Fig. 11 can be extended, in the event that the system according to the present embodiment has been only partially deployed. In this event, a server 700 contains a set of virtual devices 702 which are associated, on a one-for-one or other basis, with a set of legacy devices 704, the association being depicted by the arrow 716. The server 700 is also equipped with a display 718, connected to it by a line 720. The user of the camcorder 102 can discover the population of legacy devices 704 connected to the network 100 as exemplified by a discovery/announcement process depicted by a line 706. In a similar fashion, profile matching as depicted by a line 710 can serve as the basis for establishing the capability of the population of devices 704. While it is the capability of the legacy devices 704 which is actually of interest to the user of the camcorder 102, as exemplified by an actual information path 712, the discovery/announcement and profile matching processes take place with the set of corresponding virtual devices 702. A dashed line 722 ilustrates how the virtual devices can also provide data transfer and processing if required. The user of the camcorder 102 can use the server 700 and the associated server display 718 as a basis for constructing the desired services, and using the desired devices.

Fig. 17 depicts an example of the preferred embodiment, which should be considered in conjunction with Fig. 18 which shows a browser graphical user interface (GUI) 1508 in more detail. A local network 1500 is shown, to which is connected a local printer 1502, and a remote printer 1504 which is accessible on the Internet. A user plugs a laptop (not shown) into the local network. Running on the laptop is a browser 1506. The browser graphical user interface (GUI) 1508 looks similar to the familiar Windows Explorer TM by Microsoft TM. The browser 1506 detects the printer 1502 on the local network 1500. A printer icon 1600, possibly read from a device profile at discovery/announcement, appears at the top of the browser GUI 1508 (see Fig. 18). Next the user plugs in a digital camera 1510. A camera icon 1602 appears on the browser window 1508. The user clicks it. This causes a display of information 1604 about the camera in the left pane of the browser window 1508. The description of the camera in XML is displayed with a style sheet. Some of the items are visibly active. One item 1606, says "thumbnails". The user clicks on this and the right pane 1608 displays thumbnails of the pictures eg 1610.

The user wants to print one of the pictures 1610. He clicks on the picture 1610 and drags it to the printer icon 1600 in the icon bar. To give extra control of the operation he does this with the right mouse button (as in Windows ExplorerTM). A menu appears (not shown). The user leaves most options unchanged from the default, however he clicks the "secure pickup" checkbox, then "OK". This sequence of events activates a print job whereby the selected picture 1610 is public-key encrypted, and sent to the printer 1502 for printing. The printing process is, however, suspended until the user later goes to the printer and inserts a smartcard or Java RingTM with matching private key, thus reinstating the print process by providing a corresponding private key in a secure manner ie only in his actual physical presence.

The described embodiment assumes that devices connect to a local IP network 1500. Before the browser 1506 can announce itself and look for other network service, eg the printer 1502, the browser 1506 establishes itself on the network. It does so by listening for incoming service requests on some port. Incoming service requests are HTTP requests, so a service is uniquely identified by an HTTP URL. So for example if the user's machine upon which the browser 1506 is running has IP address 10.9.8.7 and the browser 1506 on the machine listens on port 9090, then the URL HTTP://10.9.8.7:9090 identifies the browser so that other devices and/or services can contact it. Where multiple applications, or application instances, are running on one machine, and/or sharing the same ports, extensions can be added to a URL in order to uniquely identify the particulars of the case. For example, a file name-type of extension can be added to a URL in order to identify the application type. An example of such an extension is provided as follows:

http://10.9.8.7:9090/browser.exe

Each element in the URL extension, ie. "browser", and "exe", can identify details about the application name, and type, which are intended as a destination of a message.

Multiple instances of an application type can be uniquely identified by addition of an http query as a parameter-value pair. This is illustrated as follows:

http://10.9.8.7:9090/browser.exe?instance=1

The term "instance=1" represents a parameter and its associated value, these being used to identify a particular instance of an application. It is noted that the aforementioned methods may not, in a particular instance, all be required, depending on the particular application type or standardisation being employed.

Accordingly, a unique identity is assigned to each browser window.

The principles of the preferred method described herein have general applicability to automatic network configuration and communication establishment. However, for ease of explanation, the steps of the preferred method are described with reference to a particular set of image devices and printers. However, it is not intended that the present invention be limited to the described method.

The aforementioned preferred embodiments utilise particular methods of inter-device capability negotiation. Furthermore, particular instances of discovery are described, where discovery relates to the ability of a device connected to a network to "discover" and identify other devices connected thereto. There are other variations of the preferred methods of inter-device negotiation, which use different negotiation processes without departing from the spirit or scope of the invention. A similar comment applies to the aspects of security incorporation, and device discovery. Furthermore, one or more of the steps of the preferred methods may be performed in parallel rather than sequentially.

Industrial Applicability

It is apparent from the above that the embodiments of the invention are applicable to the telecommunications, data processing and distributed computing industries.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention. The described embodiments should be considered to be illustrative and not restrictive.

Appendix

A number of XML syntax, command, and data examples are now provided.

Element Name	Context	Attributes	Description
`Message`	*top*	`From:` sender URL `to:` recipient URL `time:` time the message was sent `xmlns:` default namespace `xmlns:`*`device`*: device namespace	A Message contains a sequence of commands.

Table A1: Example Basic Message Syntax

Element Name	Context	Attributes	Description
`new`	`Message`		New service is available. The content of this tag is the service description.
`delete`	`Message`		Service is no longer available.
`describe`	`Message`		Request a service description.
`update`	`Message`		An updated or new service description (in answer to a `describe`, for example). The content of this tag is the service description. It could also be only *part* of its description, in case of a real update.
`getData`	`Message`		Request for data. The content of this tag depends on the service; it is usually an excerpt from the service description.
`data`	`Message`		Data sent by a service. The content of this tag is typically an `image` or a `document`.
`set`	`Message`		Set a service parameter.
`message`	`Message`	`status`: "SUCCESS" or "ERROR"	Provides feedback information to a command. Content of this tag is a textual message. For example, a printer could answer "3 pages printed", or a camera could answer "command not supported".

Table A2: Example Basic Command Set

The following elements can occur inside a command: they define the basic data that services can exchange.

Element Name	Context	Attributes	Description
image	Data	Name format: "JPEG" or "GIF"	An image. The content of this tag is the actual data for the image (a BASE64 element) and a command (getData) to get the *reference* image.
BASE64	Image		Some binary data (currently used for images) encoded in base64.
Document	Data	name format: "XML"	A text document. The content of this tag is the actual document, which must be valid XML.

Table A3: Example Basic Data

5 The following are examples of service or device descriptions. Some attributes have been selected to match Jini (TM) definitions for convenience and these are indicated with a ^{J} in the table.

Element Name	Context	Attributes	Description
device	new, update	url xmlns: default namespace xmlns:*device*: device namespace	
identity	device	name ^{J} manufacturer^J vendor ^{J} version ^{J} model ^{J} serialNumber ^{J} class owner	
location	device	slot floor ^{J} room ^{J}	The location tag is intended to provide information about the physical location of a

		building J map1	service in a single building or on a small, unified campus. The map attribute should refer to a map image that can be displayed in the device browser.
address	device	street J organisation J organisationUnit J locality J stateOrProvince J postalCode J country J map	The address tag provides information about the physical location of a service in a large, geographically distributed organisation.
status	device	severity J: "error", "warning", "notice", or "normal" value: "idle", "processing" or "stopped" msg: a message	Report the status of a service.
comment	device	comment J	A comment about a service. (for example, "this printer gets jammed if fed with overheads!")
content	device		The data held in the device. For example, the content for the camera is the pictures that were taken.
Commands	device		The list of commands this device can accept

Table A4: Example Basic Descriptions

In addition, all the elements above might have the following attributes:

- jini_class is used to indicate which Jini class the element corresponds to.
- text_value is a textual representation of the element, used when the device browser displays a list of devices in a tabular format. It can have one of the following values:
  - #ATTR: get the textual representation from the element attributes

- #XML: the textual representation is the XML form of the element
- #CONTENT: get the textual representation from the text content
- other: the given string is the textual representation

Examples of camera specific, and computer specific, commands and elements are provided in the following tables.

Element Name	Context	Attributes	Description
camera: getData	Message	name: Image name format: "GIF" or "JPEG"	A request to get an image. This tag is always empty.
camera: thumbnails	content, getData	base : thumbnails directory refbase :reference images directory start : first thumbnail end : last thumbnail suffix: file suffix format: "GIF" or "JPEG"	The images currently in the camera, represented by thumbnails (small representation of the images). This tag is always empty.
camera: images	content, getData	base : image directory start : first image end : last image suffix: file suffix format: "GIF" or "JPEG"	The images currently in the camera. This tag is always empty.

Table A5: Example Basic Camera-Specific Commands & Elements

Element Name	Context	Attributes	Description
computer: document	content, getData	name: name or description thumb : URL to a thumbnail image of the document href : URL to the document format : "XML"	A document. This tag is always empty.

Table A6: Example Basic Computer-Specific Commands & Elements

Claims:

1. A method of conducting a secure process between a first device and a second device, said method comprising steps of:

inputting security key information into the first device;

establishing said process between the first device and the second device;

suspending the process at a stage prior to completion thereof; and

enabling the suspended process to proceed to completion dependent upon inputting corresponding key information into the second device.

2. A method according to claim 1, whereby said security key information is one of a physical key and an electronic key.

3. A method according to claim 1, whereby said security key is an electronic public key, and said corresponding key is the physical key being a private key.

4. A method of establishing secure communications substantially as described herein according to any one of the embodiments, as that embodiment is described in the accompanying drawings.

DATED this First Day of May 2000

Canon Kabushiki Kaisha

Patent Attorneys for the Applicant

SPRUSON & FERGUSON

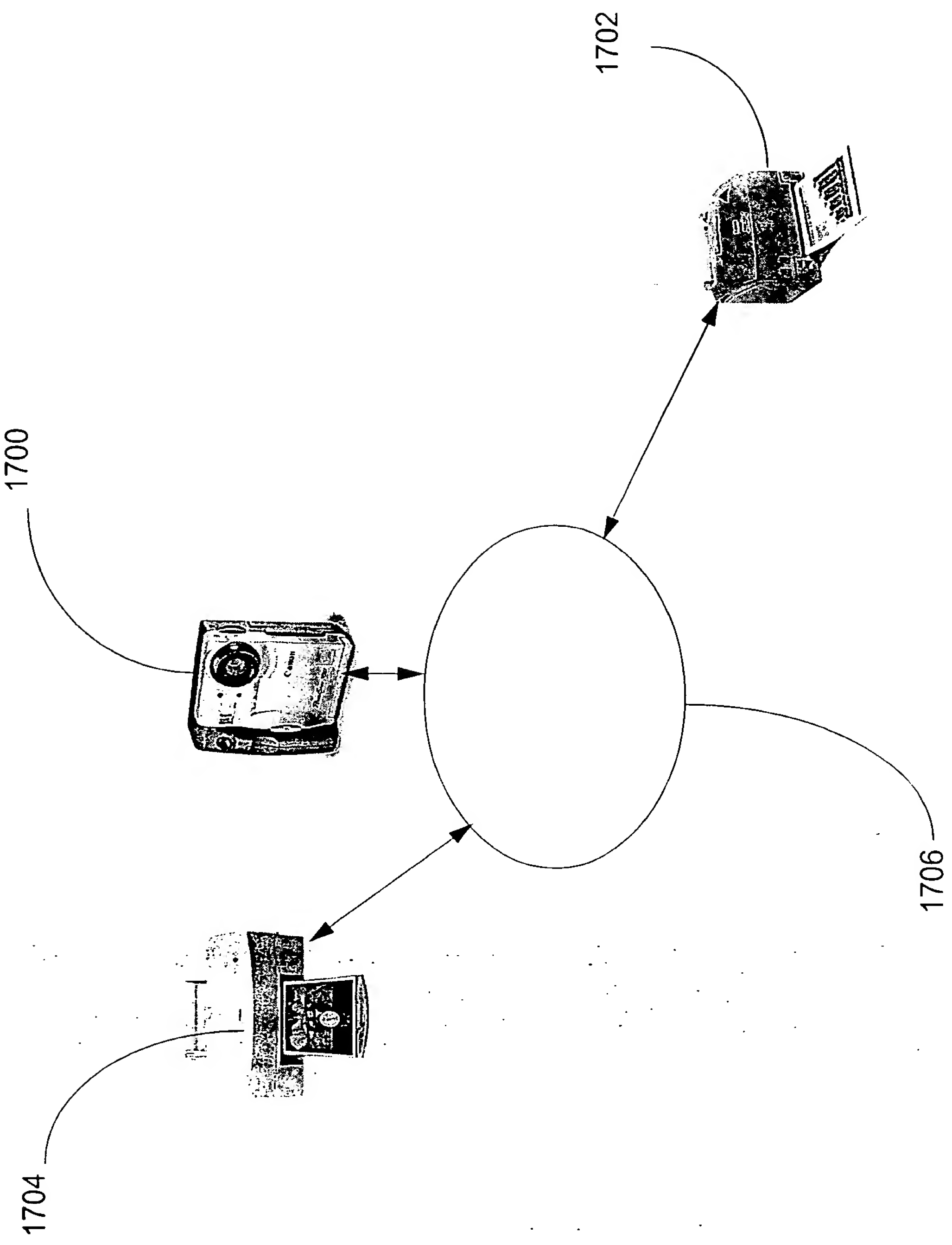


Fig. 1

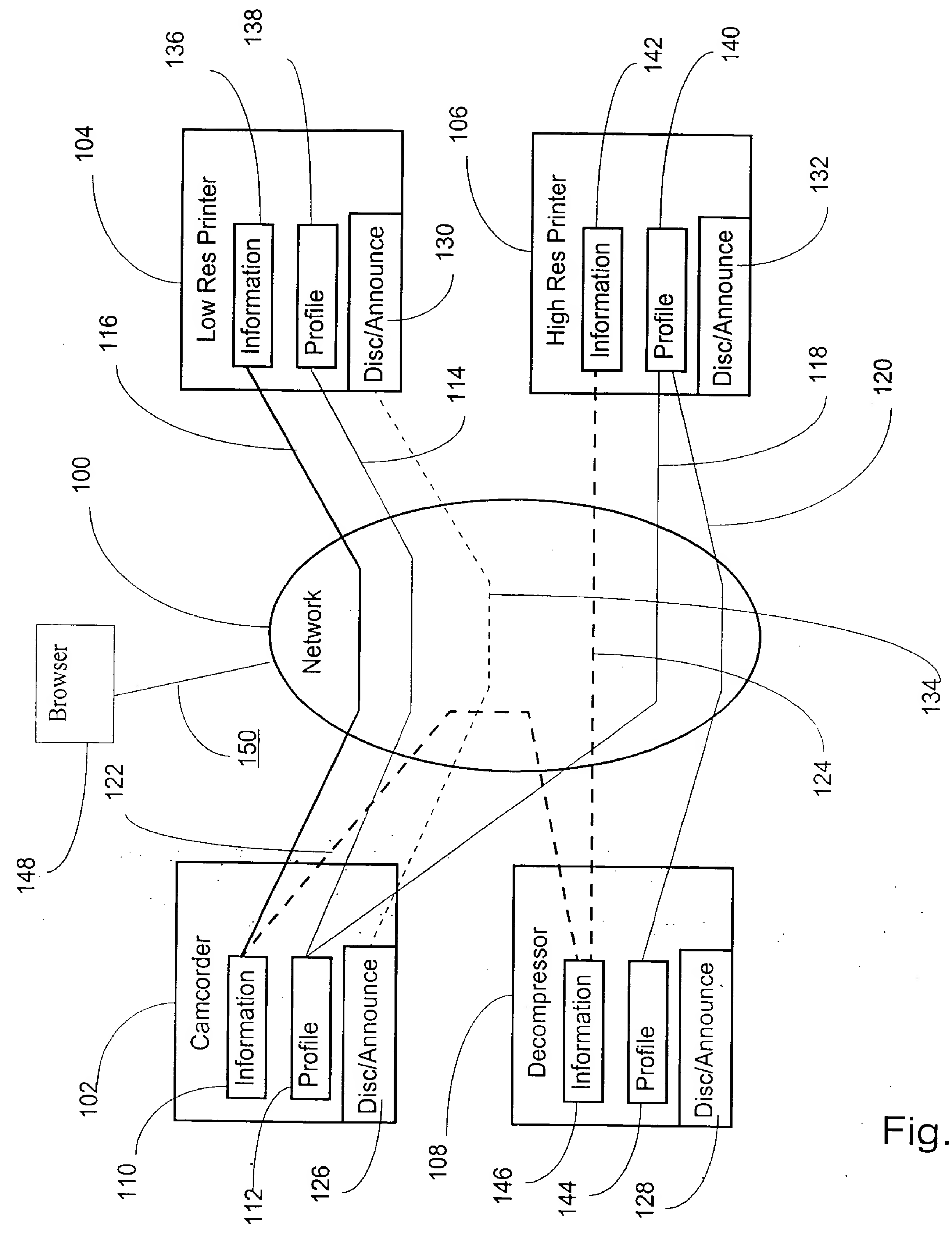

Low Res Printer
Information
Profile
Disc/Announce
High Res Printer
Information
Profile
Disc/Announce
Network
Browser
Camcorder
Information
Profile
Disc/Announce
Decompressor
Information
Profile
Disc/Announce
100
102
104
106
108
110
112
114
116
118
120
122
124
126
128
130
132
134
136
138
140
142
144
146
148
150

Fig. 2

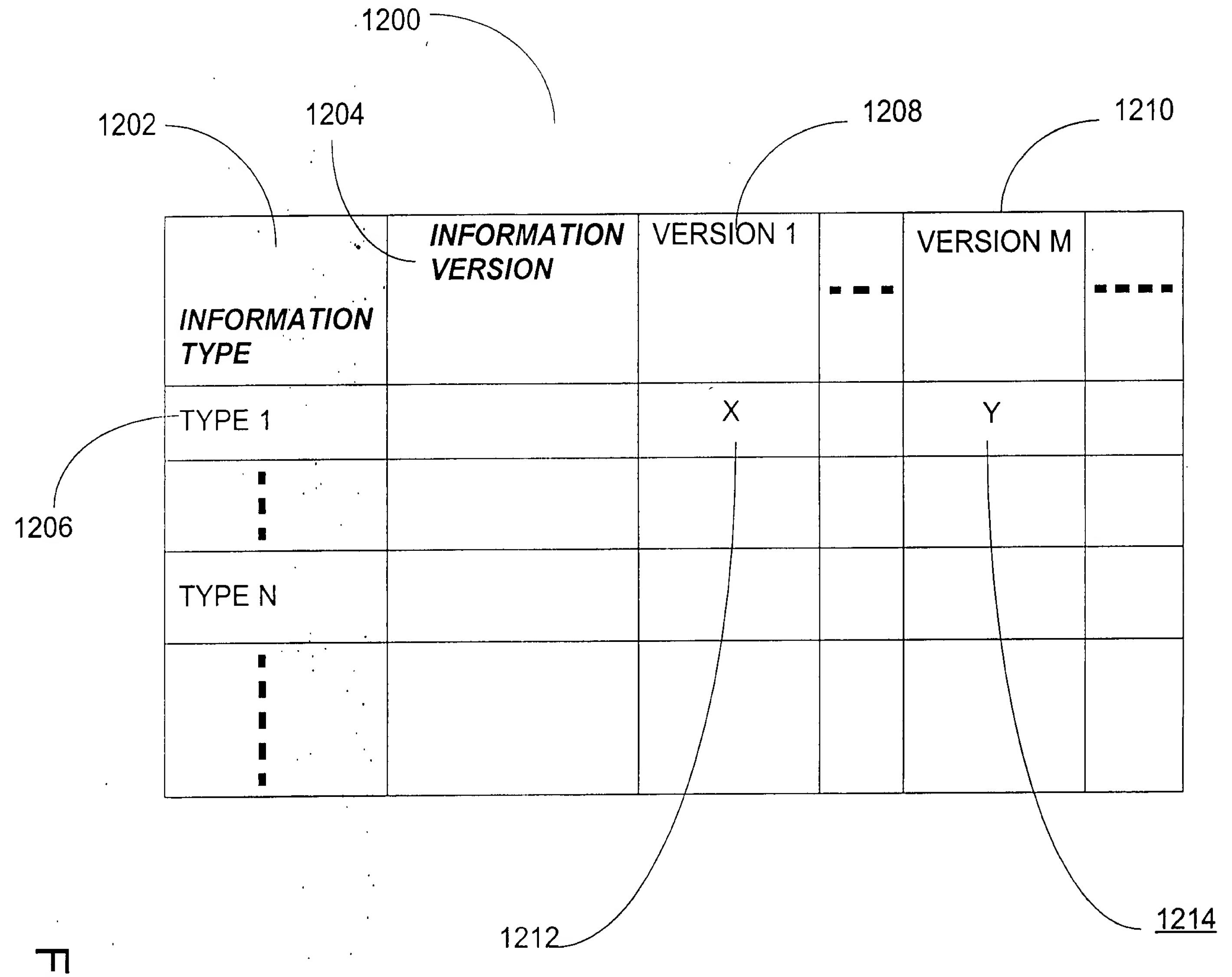

1200
1202
1204
1208
1210
INFORMATION TYPE
INFORMATION VERSION
VERSION 1
VERSION M
TYPE 1
TYPE N
X
Y
1206
1212
1214

Fig. 3

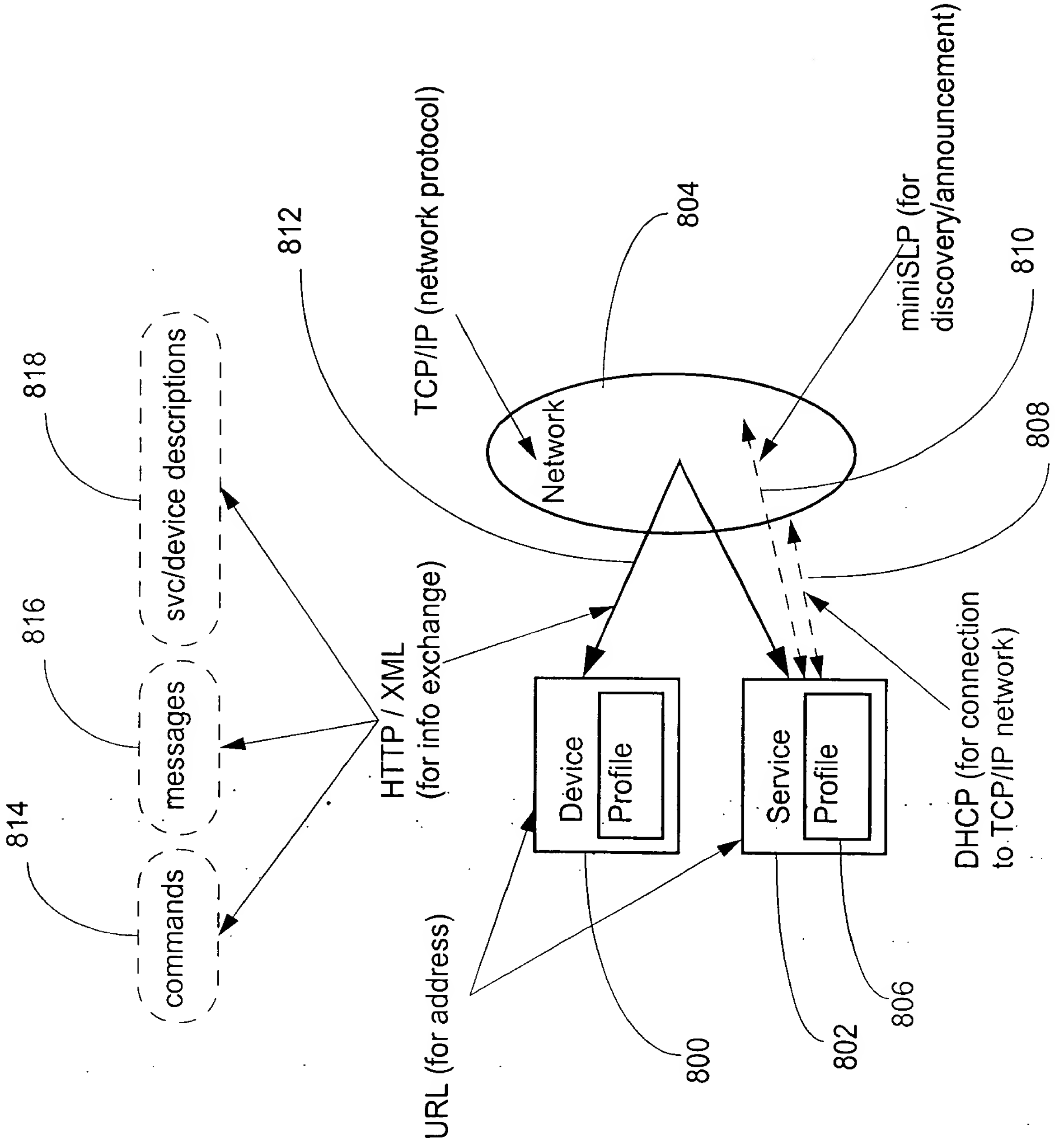

commands
814
messages
816
svc/device descriptions
818
HTTP / XML
(for info exchange)
URL (for address)
Device
Profile
800
Service
Profile
802
806
Network
804
812
TCP/IP (network protocol)
miniSLP (for
discovery/announcement)
810
808
DHCP (for connection
to TCP/IP network)

Fig. 4

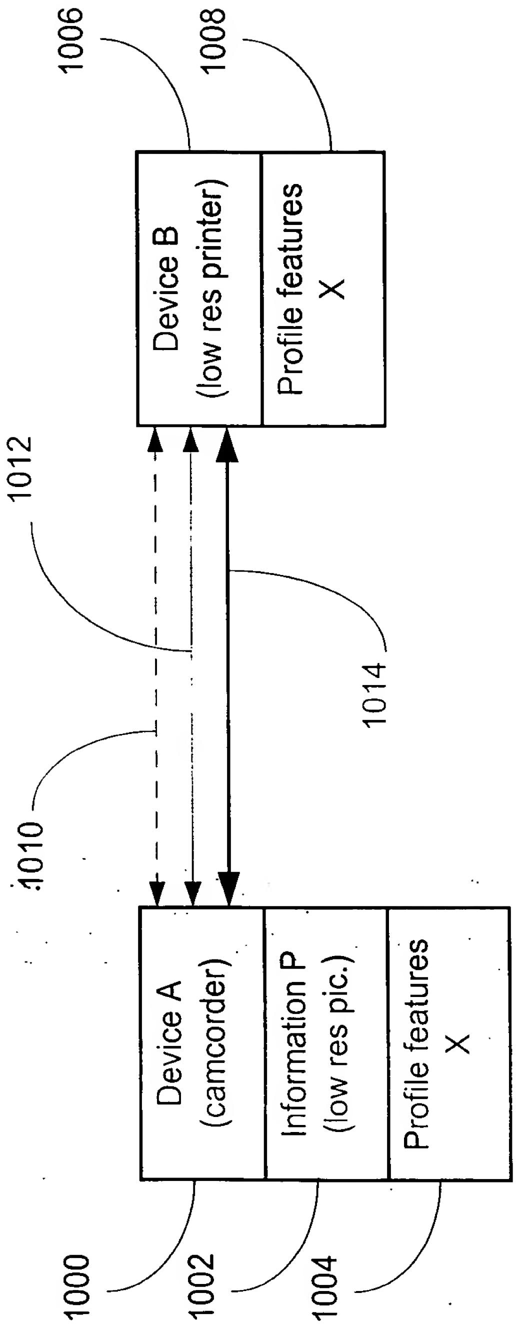
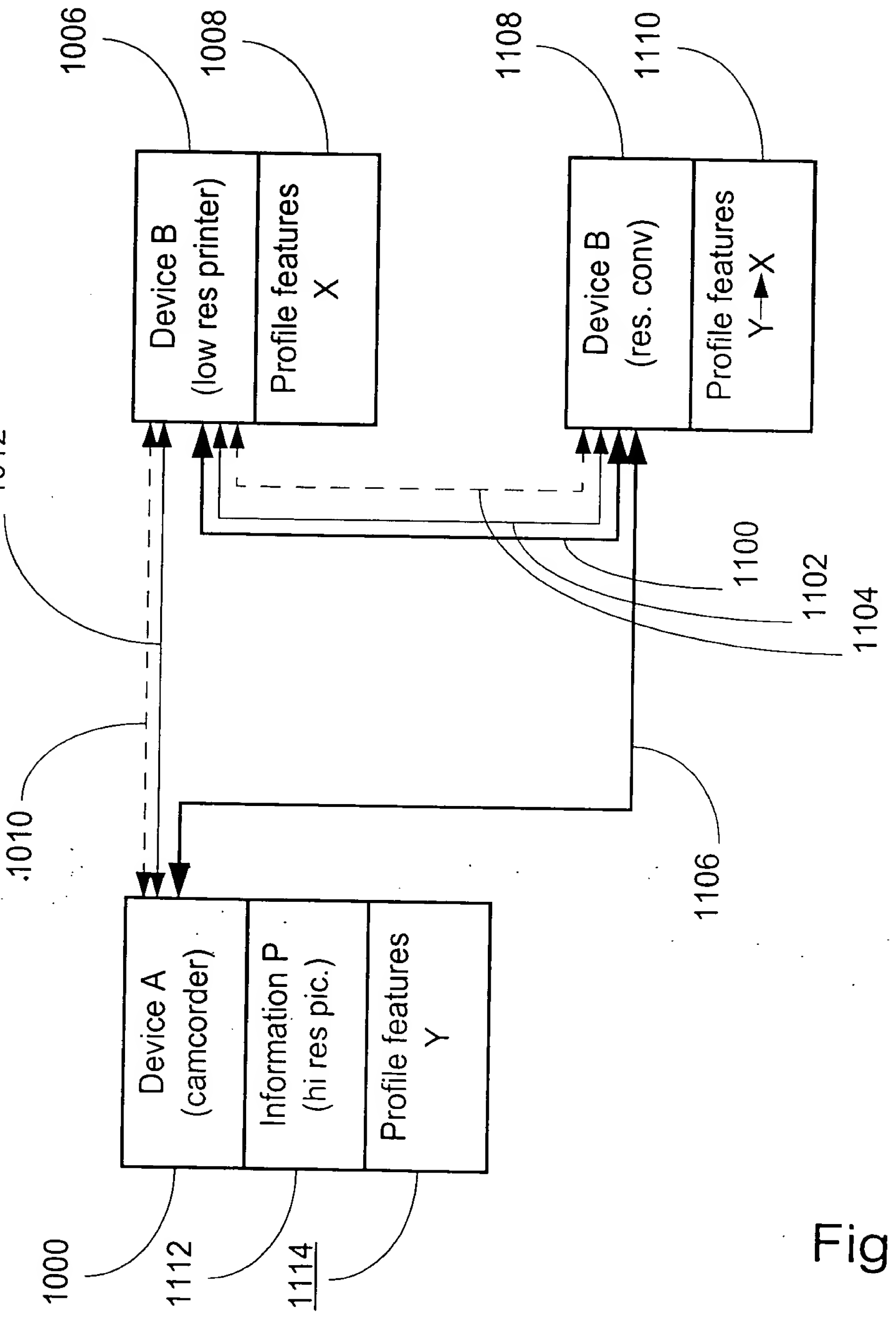

1006
1008
Device B
(low res printer)
Profile features
X
1012
1014
1010
Device A
(camcorder)
Information P
(low res pic.)
Profile features
X
1000
1002
1004

Fig. 5



Device A (camcorder)
Information P (hi res pic.)
Profile features Y
Device B (low res printer)
Profile features X
Device B (res. conv)
Profile features Y→X
1000
1112
1114
1006
1008
1108
1110
1010
1012
1100
1102
1104
1106

Fig. 6

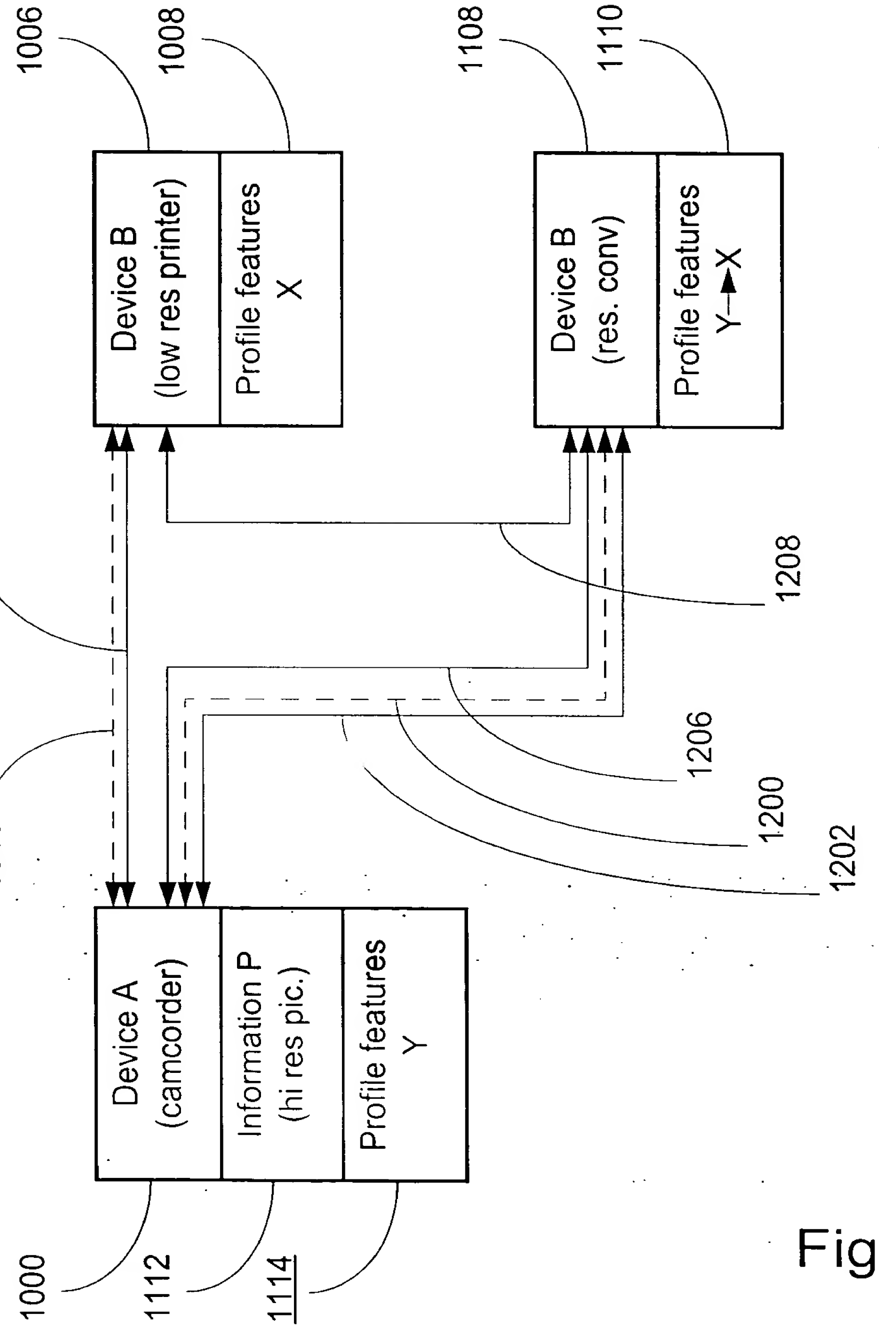
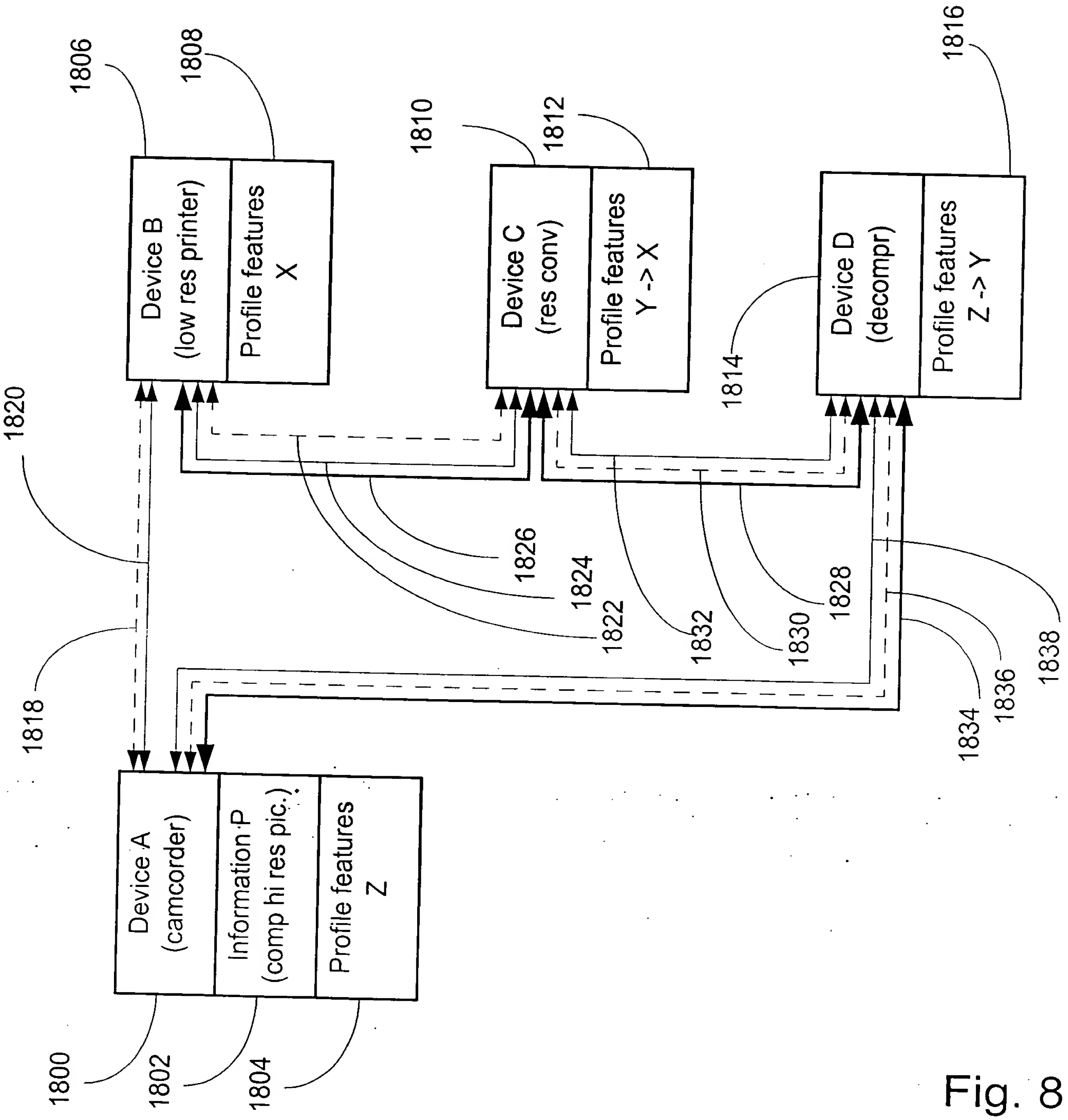

Device A (camcorder)
Information P (hi res pic.)
Profile features Y
Device B (low res printer)
Profile features X
Device B (res. conv)
Profile features Y→X
1000
1112
1114
1006
1008
1108
1110
1010
1012
1200
1202
1206
1208

Fig. 7



Device A (camcorder)
Information P (comp hi res pic.)
Profile features Z
Device B (low res printer)
Profile features X
Device C (res conv)
Profile features Y -> X
Device D (decompr)
Profile features Z -> Y
1800
1802
1804
1806
1808
1810
1812
1814
1816
1818
1820
1822
1824
1826
1828
1830
1832
1834
1836
1838

Fig. 8

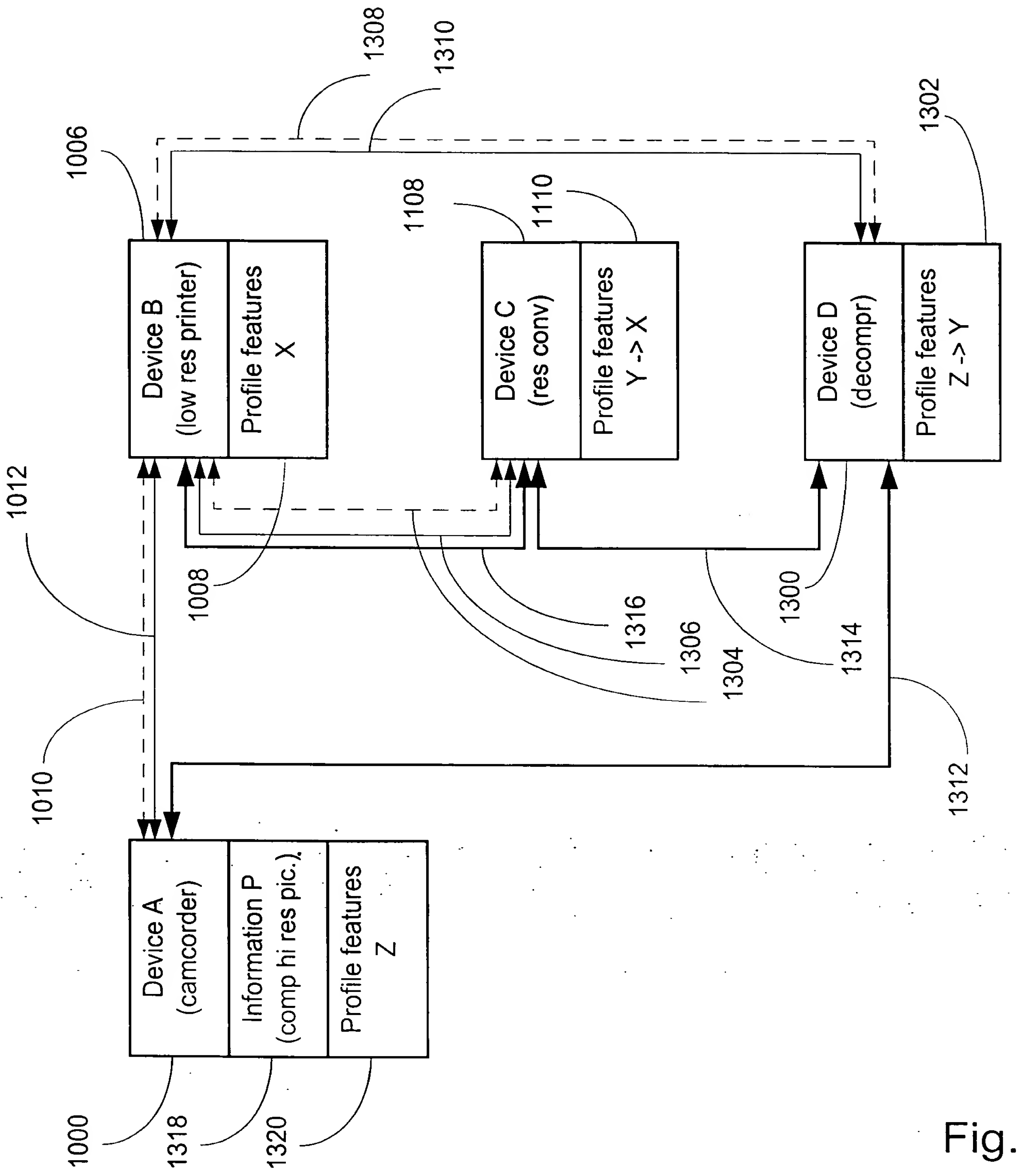

Device A (camcorder)
Information P (comp hi res pic.)
Profile features Z
Device B (low res printer)
Profile features X
Device C (res conv)
Profile features Y -> X
Device D (decompr)
Profile features Z -> Y
1000
1318
1320
1010
1012
1006
1008
1108
1110
1300
1302
1304
1306
1308
1310
1312
1314
1316

Fig. 9

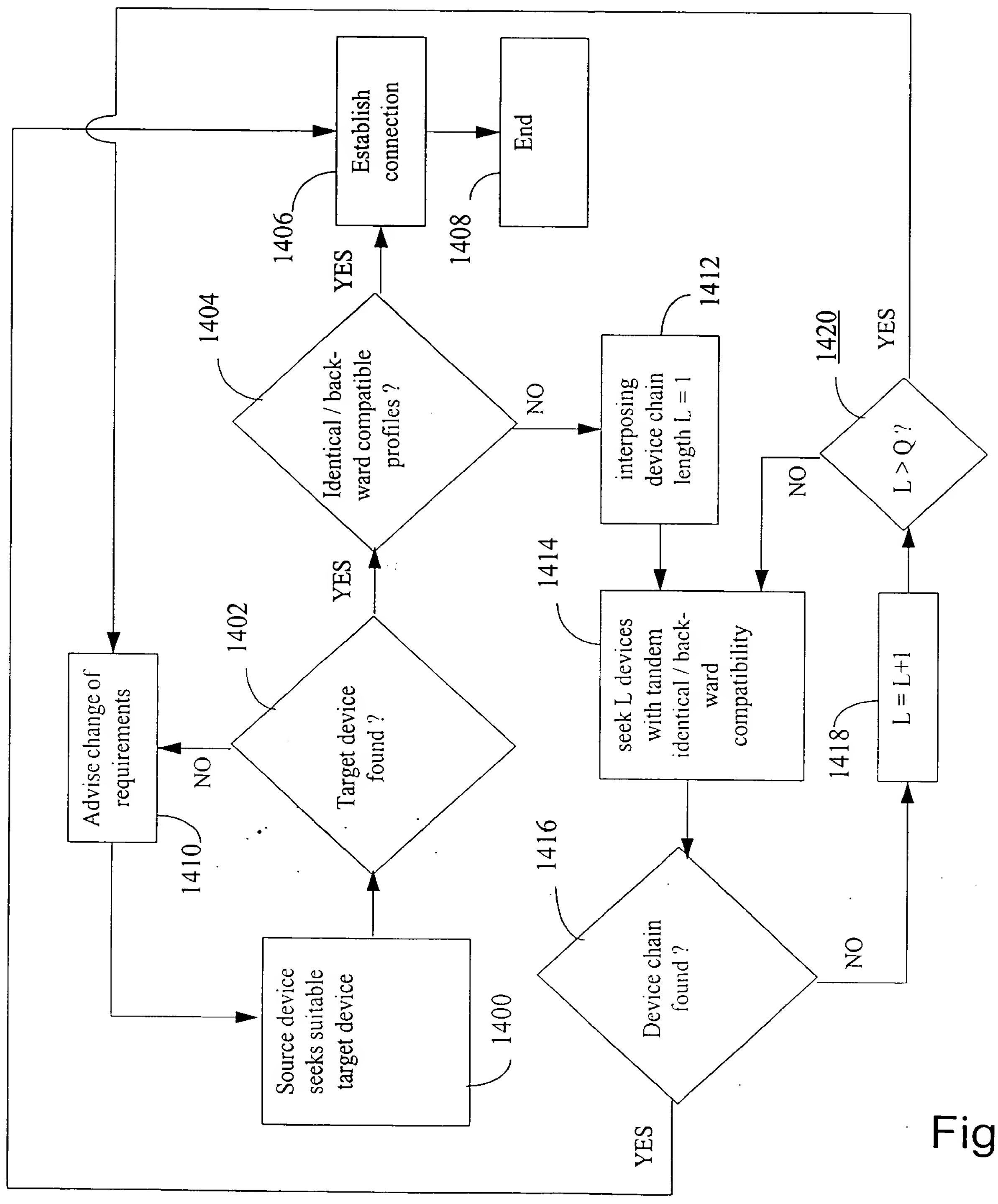
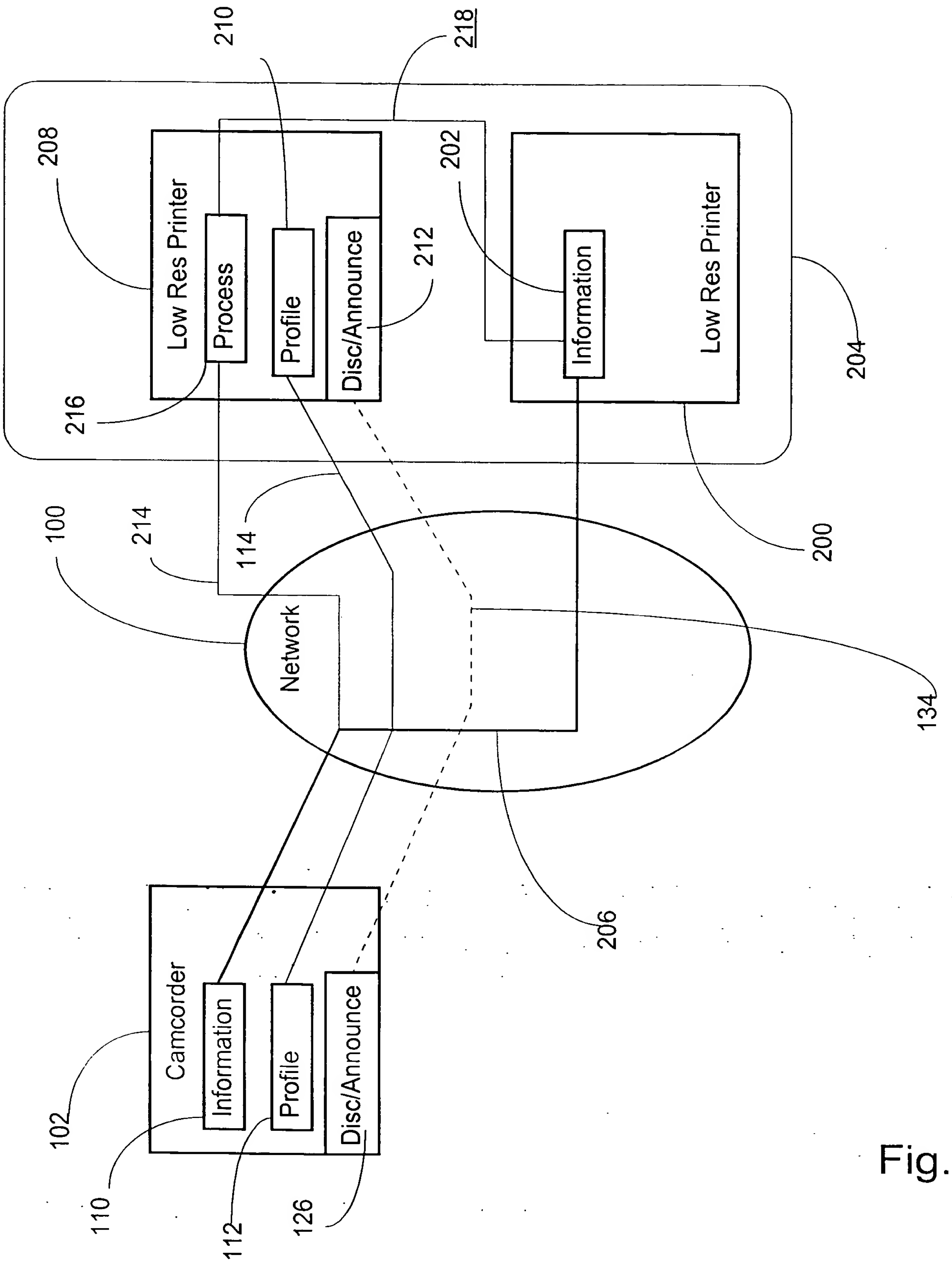

Source device seeks suitable target device
1400
Target device found ?
1402
NO
YES
Advise change of requirements
1410
Identical / back-ward compatible profiles ?
1404
YES
NO
Establish connection
1406
End
1408
interposing device chain length L = 1
1412
seek L devices with tandem identical / back-ward compatibility
1414
Device chain found ?
1416
YES
NO
L = L+1
1418
L > Q ?
1420
YES
NO

Fig. 10



218
210
208
202
Low Res Printer
Process
Profile
Disc/Announce
212
Information
Low Res Printer
204
216
200
100
214
114
Network
134
206
Camcorder
Information
Profile
Disc/Announce
102
110
112
126

Fig. 11

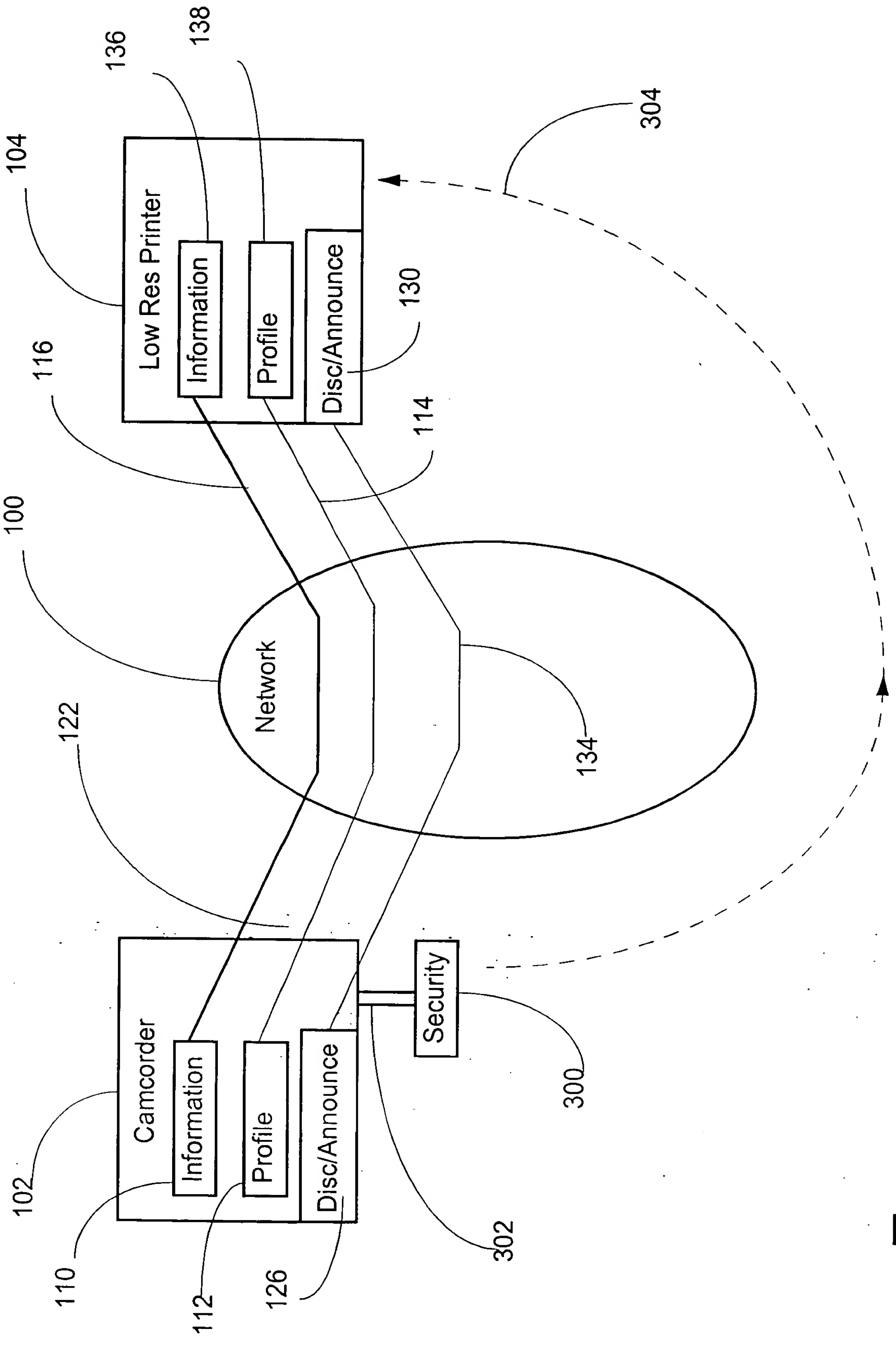
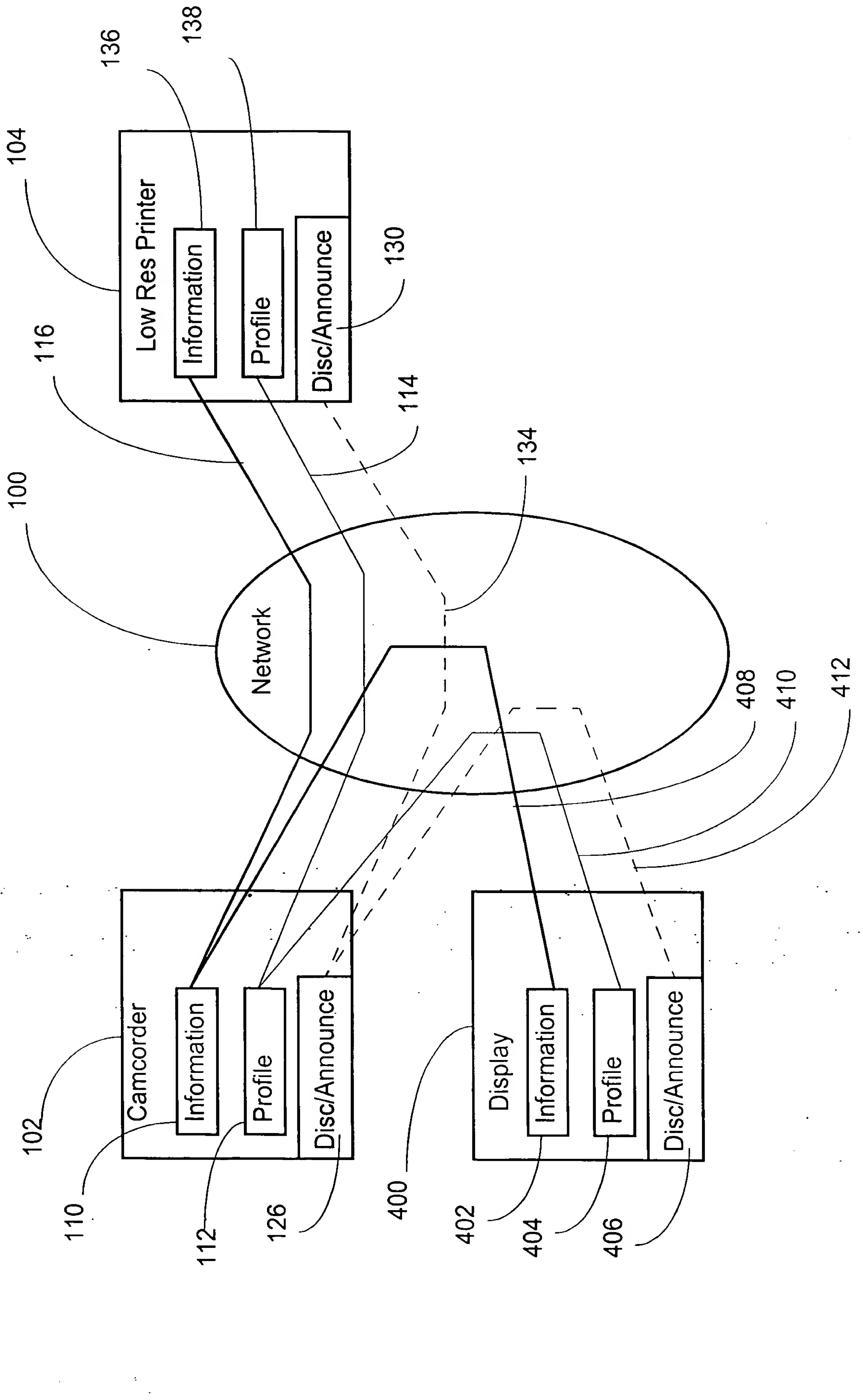

104
136
138
304
Low Res Printer
Information
Profile
Disc/Announce
130
116
114
100
Network
134
122
Camcorder
Information
Profile
Disc/Announce
Security
300
302
102
110
112
126

Fig. 12



Low Res Printer
Information
Profile
Disc/Announce
Network
Camcorder
Information
Profile
Disc/Announce
Display
Information
Profile
Disc/Announce
104
136
138
130
116
114
100
134
408
410
412
102
110
112
126
400
402
404
406

Fig. 13

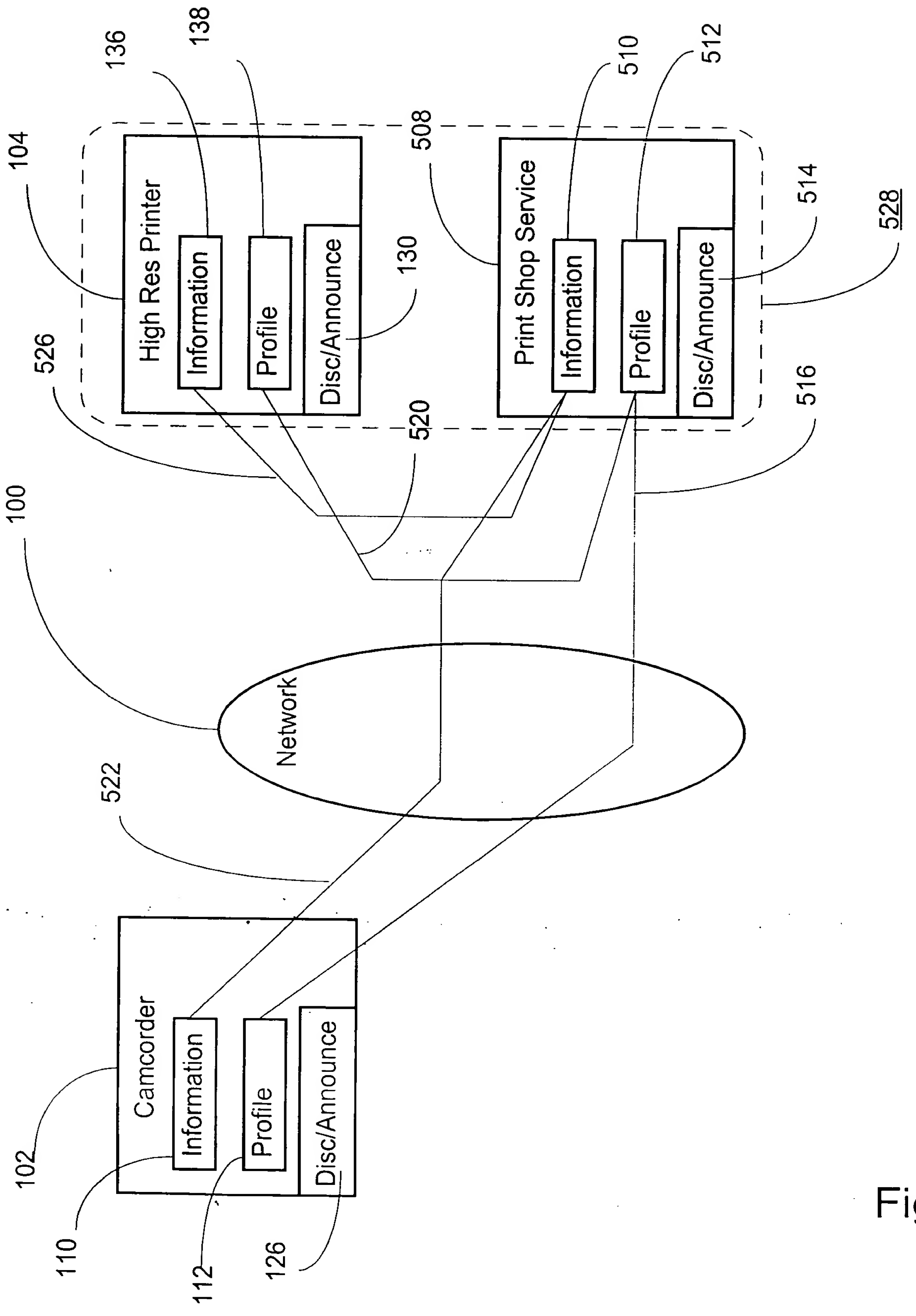

High Res Printer
Information
Profile
Disc/Announce
Print Shop Service
Information
Profile
Disc/Announce
Network
Camcorder
Information
Profile
Disc/Announce
104
136
138
130
526
508
510
512
514
528
520
516
100
522
102
110
112
126

Fig. 14

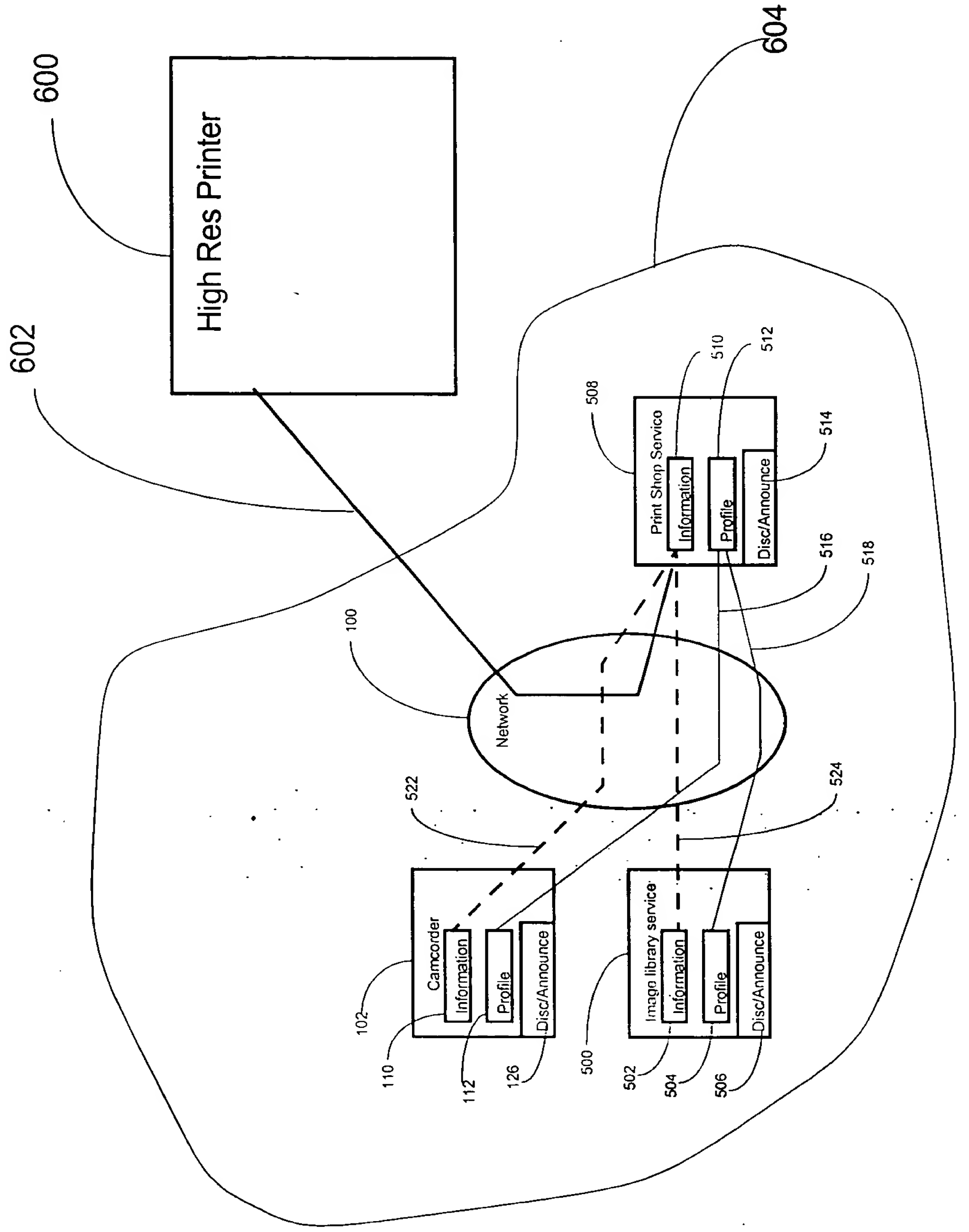

High Res Printer
600
602
604
Print Shop Service
Information
Profile
Disc/Announce
508
510
512
514
516
518
Network
100
522
524
Camcorder
Information
Profile
Disc/Announce
102
110
112
126
Image library service
Information
Profile
Disc/Announce
500
502
504
506

Fig. 15

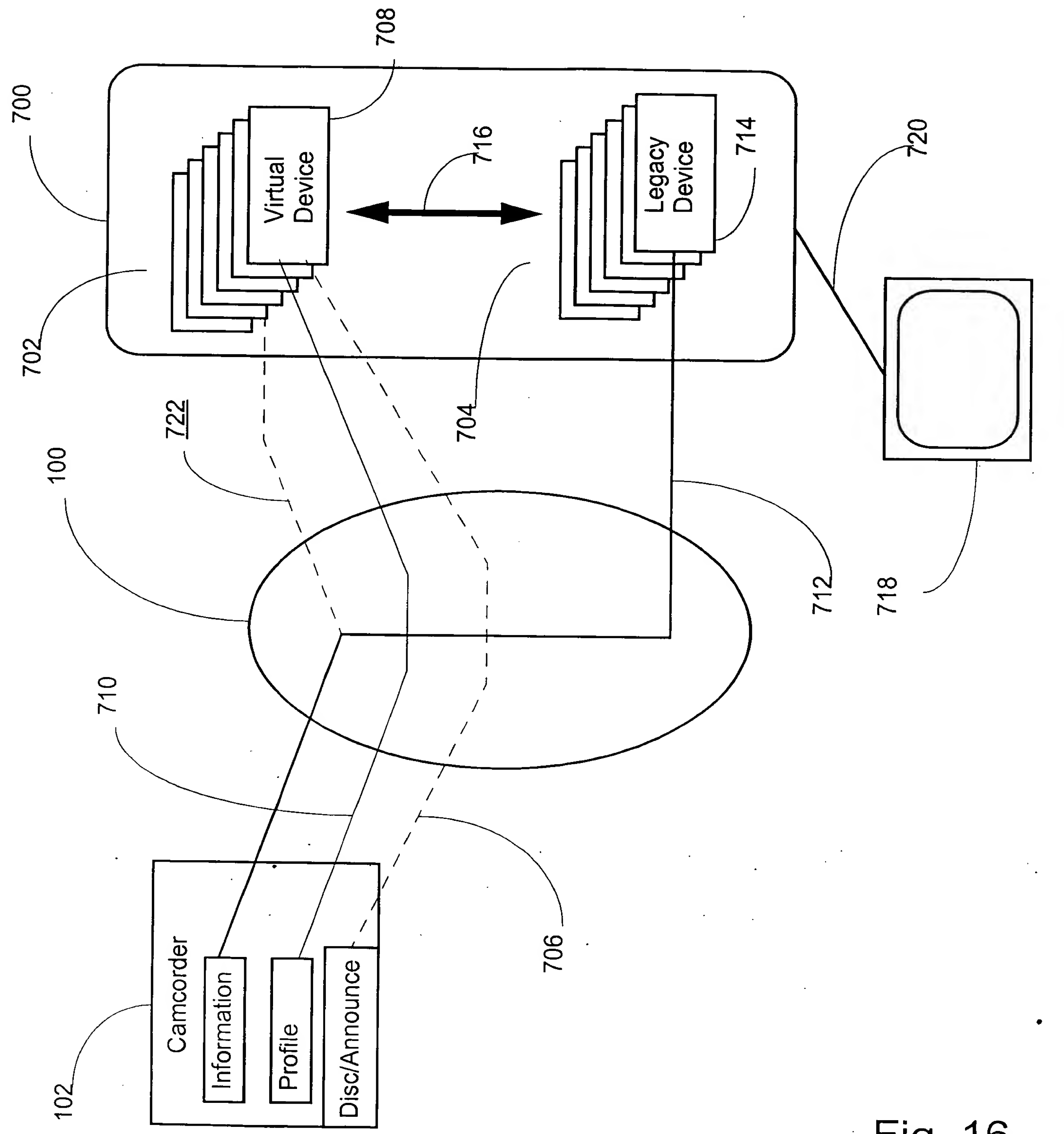

700
708
Virtual Device
716
Legacy Device
714
720
702
722
704
100
712
718
710
706
102
Camcorder
Information
Profile
Disc/Announce

Fig. 16

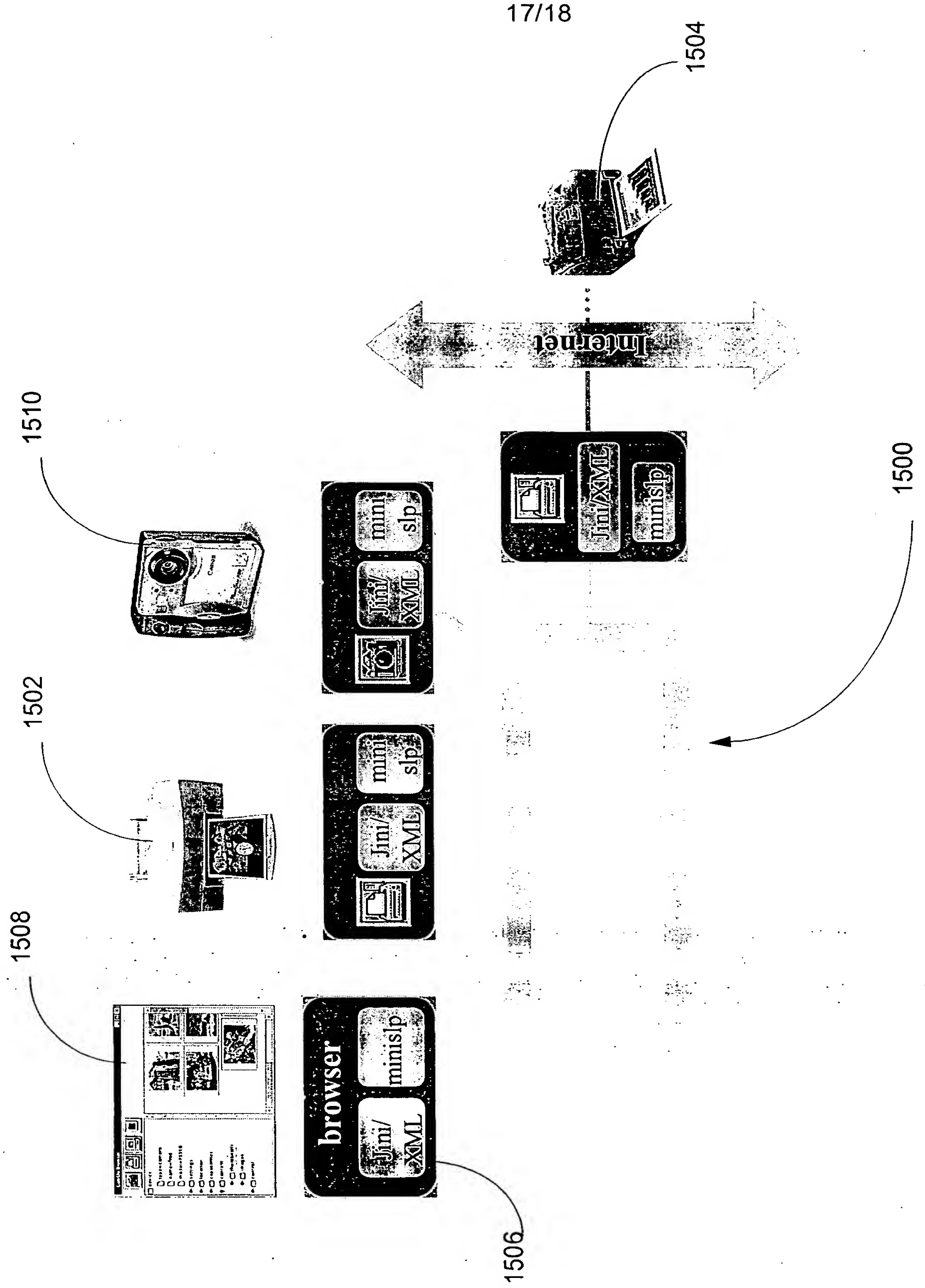

1504
Internet
1510
1502
1508
Jini/XML
minislp
Jini/ XML
mini slp
Jini/ XML
mini slp
browser
Jini/ XML
minislp
1500
1506

Fig. 17

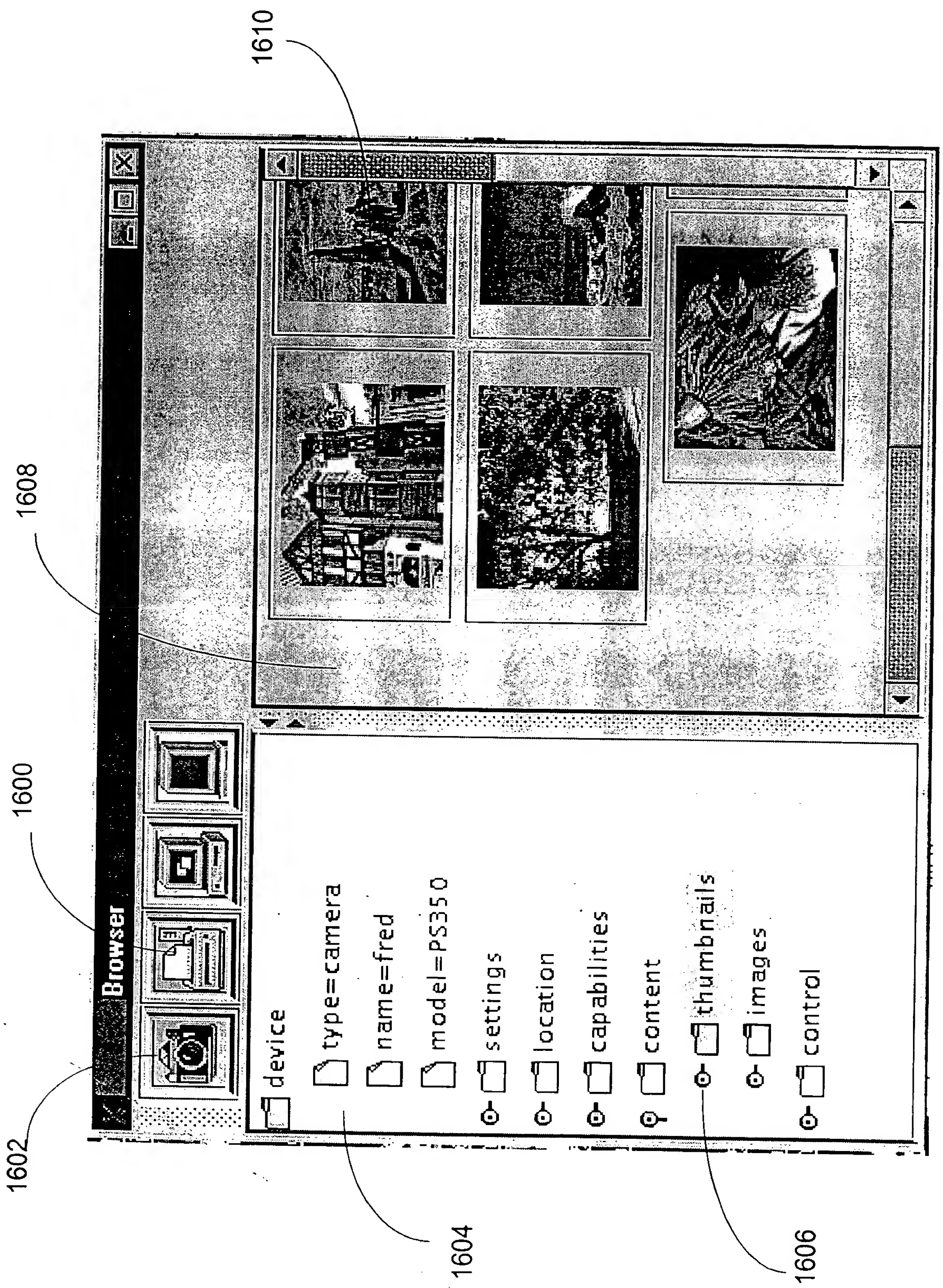

1600
1602
1604
1606
1608
1610
Browser
device
type=camera
name=fred
model=PS350
settings
location
capabilities
content
thumbnails
images
control

Fig. 18